Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Uma Nova Abordagem de Mapeamento e Localização Simultâneo Planar utilizando sensores RGB-D Baseada em Subtração de Objetos

Leonardo Angelo V. de Souto

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Computação Distribuída | Sinais, Robótica e Visão

Tiago Pereira do Nascimento
(Orientador)

João Pessoa, Paraíba, Brasil

S726u Souto, Leonardo Angelo V. de.
Uma nova abordagem de mapeamento e localização simultâneo planar utilizando sensores RGB-D baseada em subtração de objetos / Leonardo Angelo V. de Souto.- João Pessoa, 2016.
60f. : il.
Orientador: Tiago Pereira do Nascimento
Dissertação (Mestrado) - UFPB/CI
1. Informática. 2. Computação distribuída. 3. Mapeamento 3D. 4. Planar SLAM. 5. Robôs móveis.

UFPB/BC CDU: 004(043)

Resumo

Este trabalho tem como objetivo apresentar uma nova abordagem em RGB-D Planar SLAM, aqui chamada SLAM Baseado em Subtração de Objetos (OS-SLAM), utilizando apenas sensores RGB-D de baixo custo, como o Microsoft Kinect. Essa abordagem subtrai objetos do ambiente 3D e cria um mapa 3D limpo, projetando planos estimados sem os objetos identificados e encontrados ao longo do procedimento de mapeamento. Para validar esta abordagem, foram realizados quatro experimentos. Três deles em um ambiente real e um com um conjunto de dados abertos comparando-os com outros algoritmos do estado-da-arte. Os algoritmos foram aplicados a uma plataforma turtlebot 2, que tem um sensor RGB-D como seu sensor principal. Os resultados demonstraram a eficiência na geração de mapas limpos e precisos utilizando o algoritmo OS-SLAM.

Palavras-chave: Mapeamento 3D, Planar SLAM, Robôs Movéis.

Abstract

This paper aims to present a novel approach in planar RGB-D SLAM, here called Object Subtraction SLAM (OS-SLAM), using only low-cost RGB-D sensors such as the Microsoft Kinect. Our approach subtracts objects from the 3D environment and creates a clean 3D map projecting estimated planes without identified objects encountered along the mapping procedure aiming to increase accuracy. To validate our approach we performed four experiments, three in a real environment and one with an open dataset comparing them with other state-of-the-art algorithms. The algorithms were applied to a TurtleBot 2 platform which has a RGB-D sensor as its main sensor. The results demonstrated the efficiency on generated a clean and accurate map using the OS-SLAM algorithm.

Keywords: 3D Mapping, Planar Slam, Mobile Robots.

Agradecimentos

Agradeço, primeiramente, a Deus, por ter me concedido saúde e capacidade para realização deste trabalho. Toda honra e toda glória seja dada a Ele. Aos meus pais, Adalberto Alves e Claúdia Germana, por todo apoio, dedicação e orientação destinadas a minha pessoa durante toda a vida, bem como aos meus queridos irmãos, Lisandra Raquel e Felype Augusto, que os amo. A minha querida esposa, Marina Maciel Sitônio, que me apoiou e me ajudou todos os dias desse período: obrigado, sem você nada disso estaria acontecendo. Ao meu sogro, Marcelo Xavier Sitônio, e minha sogra, Simone dos Santos Maciel, que são exemplos de dedicação e disciplina para minha pessoa, além do apoio e dedicação que destinam a minha família, em que tenho, juntamente com os meus pais, como exemplos, apoio e fonte de experiência para minha vida e espero nem tão cedo perder toda essa base. A minha grande amiga, Talita Stael, por me obrigar a fazer a seleção do mestrado um dia antes do término do período de inscrição, e por acreditar sempre no meu potencial. Ao meu orientador, Professor Tiago Pereira do Nascimento, por todo o tempo gasto de bom grado em minha orientação, não somente acadêmica, uma vez que aprendi muito ao decorrer desses dois anos de trabalho, em todos os aspectos de minha vida.

Conteúdo

Lista de Símbolos

SLAM : *Simultaneos Localization and Mapping*

PCL : *Point Cloud Library*

RGB-D : *Red, Green and Blue Colors*

OS-SLAM : *Object Subtraction - Simultaneos Localization and Mapping*

SURF : *Speeded Up Robust Features*

SIFT : *Scale-invariant Feature Transform*

ICP : *Iterative Closest Point*

RANSAC : *Random Sample Consensus*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste primeiro capítulo será apresentado a motivação para a realização da pesquisa, bem como os objetivos gerais e especificos além da metodologia utilizada tanto para a criação quanto para a avaliação do trabalho.

1.1 Motivação

Robôs móveis requerem o conhecimento do ambiente para as suas operações, incluindo pesquisa e tarefas de resgate, transporte e orientação. Os módulos de mapeamento e localização são caracterizados por serem fatores de destaque na comunidade robótica e são características importantes em robôs móveis autônomos [1], [2],[3] .

Mapeamento e localização simultâneos (SLAM) é uma classe de algoritmos que têm a propriedade de construir um mapa de um lugar desconhecido, enquanto o robô se move no meio ambiente. Uma vez que o mapa da área é criado, o robô é capaz de localizar-se. Os algoritmos SLAM podem resultar em um mapa 2D ou 3D [4] .

Mapeamento no campo da robótica, é caracterizado por uma interpretação dos dados adquiridos dos sensores do robô, por exemplo: scanners a laser, câmeras, sensores RGB-D, odometria, entre outros. Portanto, o mapeamento ajuda a identificar as características do meio, traduzindo-os em uma representação [5],[6] .

Na classe de mapeamento 3D de algoritmos de SLAM, há aqueles que incidem sobre o uso de um sensor RGB-D para capturar os dados do ambiente [7] . Considerando os diferentes algoritmos de SLAM na literatura, e em função da capacidade de processamento

em tempo real, o SLAM Graph-base RTAB-Map foi utilizado como base para a criação da nossa abordagem de SLAM baseada em subtração de objetos (chamada aqui de OS-SLAM), e foi feita uma comparação com esse algoritmo [8] bem como com outros algoritmos de mapeamento 3D dos últimos anos do estado-da-arte [9].

Embora o RTAB-Map seja um algoritmo de SLAM baseado em grafos, este serviu como base para a nossa abordagem. O OS-SLAM propõe um algoritmo que seu propósito é distintivo para remover os objetos (o que não for parede ou chão) do ambiente mapeado. Esta contribuição apresenta uma heurística em que os maiores planos no meio são paredes e chão, e todos os outros objetos são apenas móveis, que devem ser excluídos com o objetivo de aumentar a precisão no mapeamento. Pessoas que se deslocam também são descartadas durante a fase de mapeamento. O sistema, então, projeta os objetos no plano mais próximo, achatando-os (e, portanto, excluindo-os). Partindo desse pressuposto, essa abordagem torna-se parte da comunidade Planar-SLAM. Apresenta, também, um método para o tratamento de planos em nuvem de pontos RGB-D, por aplicação do cálculo da normal e da convexidade. Por fim, um método para minimizar e filtrar planos é aplicado, para se obter um mapeamento mais significativo em tempo real.

1.2 Contribuições

Desenvolver uma nova abordagem de Planar SLAM RGB-D capaz de mapear ambientes, sem sofrer influência de objetos ou pessoas presentes no momento do mapeamento o OS-SLAM. Fazendo uso de câmeras e plataformas mais acessíveis no desenvolvimento e testes.

Uma das contribuições que se pode elencar por parte deste trabalho é a capacidade que do algoritmo em extrair e tratar as características dos planos principais detectados nas nuvens de pontos. Resultando, assim, na apresentação de um novo método para classificação e filtragem de planos mais precisos e que venham a melhor contribuir na criação do mapa 3D.

Outra contribuição é a identificação e extração de objetos e pessoas durante o mapeamento. A abordagem apresenta uma capacidade de abstrair do ambiente, em tempo real, as áreas ocupadas por objetos móveis ou pessoas que estejam transitando no ambiente no momento do mapeamento. Possibilitando, assim, a construção de um mapa mais genérico e confiável no que diz respeito apenas a estruturas fixas e que dificilmente sofrem alteração de

localização.

Além disso, o trabalho propõe uma rotina para fusão de mapas em um cenário de multi robôs ou multi mapeamento. Através desta rotina os robôs podem mapear um mesmo local, simultaneamente, dividindo-o em áreas para cada robô, tendo um um ponto do ambiente em comum para o mapeamento. Desse modo, tem-se, ao final deste processo, será possível unir o que foi mapeado por cada um deles em um único mapa global do ambiente em questão.

A abordagem foi desenvolvida e aplicada em plataformas de robôs móveis. Tendo, assim, também como contribuição um algoritmo de Planar SLAM RGB-D já testado em ambientes reais fazendo uso do Microsoft Kinect e Turtlebot 2.

Este trabalho foi publicado na conferência Robotics Symposium (LARS) and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR), 2015 12th Latin American, como mostra a referência [39]

1.3 Metodologia

O desenvolvimento da pesquisa foi dividido em 4 etapas. No primeiro momento, foi realizada uma análise das abordagens recentes e suas funcionalidades, a fim de identificar quais as dificuldades ainda encontradas no estado-da-arte, bem como as qualidades que se tornaram essenciais no cenário atual. Na segunda fase do trabalho, desenvolve-se a abordagem OS-SLAM em sua primeira versão e realizou a avaliação de melhorias a serem desenvolvidas. Na terceira etapa da pesquisa, ocorre a adição de heurísticas na estrutura do OS-SLAM possibilitando, assim, uma versão mais completa e estável, no que diz respeito ao tratamento dos planos e criação de mapas mais genéricos sem a influência de objetos móveis. Esta será apresentada neste trabalho nas seções seguintes. Na quarta etapa, foram aplicados 4 experimentos com a plataforma TURTLEBOT 2 buscando a avaliação e validação da abordagem criada onde :

1. O primeiro experimento é realizado em um laboratório composto de diversas mesas, cadeiras, robôs e pessoas a título de comprovar a capacidade de abstração do algoritmo para com esses objetos e ver a qualidade do mapa genérico criado em comparação com um dos principais algoritmos do estado-da-arte;

2. O segundo experimento é feito em um ambiente grande, para identificar a capacidade do algoritmo no mapeamento de grandes áreas, que também possuíam a presença de objetos e pessoas ao decorrer do mapeamento;

3. No terceiro experimento, o algoritmo é avaliado num cenário multi-robôs, tendo como objetivo avaliar a capacidade de *merging* de áreas mapeadas por robôs diferentes e a qualidade do mapa global resultante;

4. O quarto experimento é uma avaliação quantitativa, onde a abordagem é testada com um *groundtruth*, e seus *datasets* que buscam recriar situações de diferentes tamanhos, velocidade de câmeras dentre outras características. Onde é também comparado com os resultados de algoritmos do estado-da-arte que fizeram uso dos mesmos *datasets*.

1.4 Estrutura da Dissertação

A dissertação continua com o Capítulo 2, que fundamenta a teoria abordada tais como Robótica Móvel, SLAM RGB-D, PCL e Planar SLAM. No Capítulo 3 é analisado o estado-da-arte. Já o Capítulo 4 apresenta a problemática desta pesquisa e a abordagem proposta. Os métodos para a coleta de dados bem como a avaliação dos resultados são encontrados no Capítulo 5. Finalmente, uma consideração final é apresentada no Capítulo 6.

Capítulo 2

Fundamentação Teórica

Neste segundo capítulo serão abordados os temas norteadores deste trabalho. Trata-se de um breve arcabouço teórico que visa dar base para um melhor entendimento das seções subsequentes.

2.1 Robótica Móvel

A robótica móvel ganhou muito espaço nos últimos anos. Isso se dá pelo ganho na flexibilidade em relação às atividades que podem ser exercidas por robôs [2]. Enquanto temos os robôs fixos bem presentes na área de fabricação, com os robôs móveis o aumento do interesse na robótica vem de todos os setores da indústria, das aplicações médicas, assistência pessoal, segurança, armazenagem e até a exploração espacial. Esse interesse surge nas capacidades apresentadas pelos robôs móveis, tais como o deslocamento ágil, grande flexibilidade e variedade de ações, possibilidade de trabalho em grupo para finalizar uma mesma atividade, dentre outras qualidades que podem ser elencadas aos inúmeros tipos de robôs que fazem parte dessa categoria [1],[4].

O espaço industrial, que por muito tempo foi dominado pelos grandes manipuladores e robôs fixos, tem seu campo redefinido pelos robôs móveis. Nesse cenário é possível observar modernas plataformas móveis, autônomas, capazes de operar em determinadas áreas sem alguma alteração ambiental ou investimento na infraestrutura existente. Essa vantagem é resultado da principal característica dos robôs móveis: superação de obstáculos e redirecionamento, através da inteligência sensorial neles equipada [6], [4].

Além disso, os robôs móveis podem, facilmente, acessar áreas inacessíveis e/ou de riscos para os seres humanos. Um exemplo bem comum são as usinas nucleares com altos níveis de radiação durante uma ameaça de um desastre. Também na área da saúde, além do auxilio em procedimentos cirúrgicos, ainda podem ser usados na entrega de medicamentos, ou nos primeiros socorros de forma mais ágil. Com todas essas funcionalidades e possibilidades que a comunidade tem identificado, o desenvolvimento, tanto de *hardware* quanto de *software*, para a robótica móvel tem aumentado continuamente [1], [10]. Na parte do *hardware* existem diferentes tipos de robôs móveis, podendo estarem clássicados como aéreos - a exemplos dos quadcópteros e drones - terrestres - nessa categoria podendo ainda serem *off-road* ou *indoor* - e robôs artrópodes (exoesqueletos) - que se assemelham com os próprios animais (ver Fig.2.1).

Figura 2.1: Quadricóptero, Hexapode e Terrestre Rusky Offroad

Neste trabalho foi usada a plataforma móvel terrestre e *indoor* Turtlebot 2 [33] (ver Fig. 2.2), porém a abordagem tem a capacidade de operar em qualquer plataforma robótica *indoor* que possua como sensor principal de visão uma câmera RGB-D. Esta é uma plataforma robótica *open source* projetada para propiciar a facilidade do ensino e da pesquisa na robótica móvel. A sua base, denominada Kobuki, possui sensores, motores e fontes de energia. Porém, sozinha não se torna funcional. De modo que se faz necessário a adição de *hardware* destinado ao processamento, que no caso do Turtlebot 2 é um netbook/notebook , e do software de integração ROS (*Robot Operating System*) [8] - um meta sistema operacional destinado a robôs móveis, onde pode ser encontrado abstração de *hardware*, controle para os dispositivos de baixo nível, troca de informações entre processos e gerenciamento de

pacotes.

O ROS também fornece ferramentas e bibliotecas para facilitar a implementação de aplicações robóticas. Desenvolvido em 2007 [8] com o nome de switchyard pelo laboratório de Inteligência Artificial de Stanford, e suportado ao projeto ESCADA Robot Stanford AI, o ROS se caracteriza por ser um *middleware* e não um sistema operacional em si. Surgiu mediante a necessidade da utilização e estudo da robótica móvel mesmo quando os robôs reais não estiverem disponíveis, facilitando assim o desenvolvimento das aplicações nesse ramo. O Turtlebot 2 Fig.2.2 está equipado com os seguintes itens de *hardware* :

1. Base Kobuki:
   (a) Bateria Lithium-Ion, 14.8V, 2200 mAh (4S1P -pequena), 4400 mAh (4S2P - grande);
   (b) Odômetro : 52 ticks/enc rev, 2578.33 ticks/wheel rev, 11.7 ticks/mm;
   (c) 12 V 1.5 cabo adapatador de alimentação para o Microsoft Kinect™;
2. Sensor 3D;
   (a) Microsoft Kinect™;
   (b) Kinect™ cabo adaptador de energia;
3. Computador :: ASUS 1215N:
   (a) Processador :: Intel® Atom™ D525 dual core;
   (b) Memoria :: 2 GB;
   (c) Grafico :: NVIDIA®ION ™ discrete graphics processor;
   (d) Internal Hard Drive :: 250 GB;
4. TurtleBot *Hardware*:
   (a) Suporte para Kinect;

(b) Estrutura do Turtlebot;

(c) Módulos de Madeira para apoio dos demais dispositivos;

5. *Software*:

(a) C ++ / Python drivers para Linux;

(b) ROS (*Robotic Operating System*);

(c) Gazebo.

Figura 2.2: TurtleBot 2 robot

2.2 Mapeamento e Localização Simultâneos

O Mapeamento e Localização Simultâneos (do inglês *Simultaneous Localization and Maping* - SLAM) se caracteriza por ser uma categoria de algoritmos aplicados ao ramo da robótica móvel que fornecem aos robôs a capacidade de mapear um ambiente desconhecido e localizar-se no mesmo [1], [2]. As pesquisas nessa área se expandiram de forma rápida, uma vez que mapear e localizar-se vem a ser um dos primeiros passos para que tenhamos um robô com um determinado grau de autonomia em suas atividades. Na figura 2.3 vemos de forma sequêncial e simplificada os passos para a autonômia de um robô:

Os algoritmos de SLAM, a priori, utilizavam-se apenas de *scanners* a *lasers* como os principais sensores de captação. Estes possuem uma grande precisão, porém, um valor muito

Figura 2.3: Passos para Navegação Autonôma

elevado, tornando as pesquisas neste âmbito restritas [11],[10]. Nos últimos anos, além destes sensores, foi relatada a possibilidade do uso de câmeras RGB-D como sensores de captação, ao exemplo do Microsoft Kinect e do Asus Xtion, estes possuem um custo bem abaixo [5], [6].

O uso de câmeras RGB-D, câmeras estéreo, dentre outros sensores de imagens, deram origem ao campo do SLAM VISUAL [7], [8], possibilitando o mapeamento de determinados ambientes tanto em 2D como em 3D (Ver Fig.2.4). A evolução rápida desta categoria de algoritmos se deu pela abrangência de situações em que podem ser aplicados, como por exemplos o mapeamento de uma determinada área de risco, tais como: interior de um prédio em chamas, uma área de vazamento químico dentre outras diversas situações que para uma pessoa se tornaria impossível de ser feita de maneira segura [3],[29].

Esse desenvolvimento rápido da área de pesquisa e os diferentes tipos de sensores de captação de imagens que surgiram, seguiram a tendência geral da evolução da robótica nos últimos anos. Esta tendência que se caracteriza pela utilização de sensores mais baratos, que em sua maioria, tendem a ser menos precisos, porém permitem uma acessibilidade maior ao público [30].

Com isso, o surgimento de diversos novos projetos de código aberto foram significantes, contribuindo de forma inestimável para a evolução das abordagens presentes no estado-da-arte. No tópico seguinte abordamos um dos projetos de código aberto que facilitaram o desenvolvimento de abordagens de SLAMs atuais [31], [29].

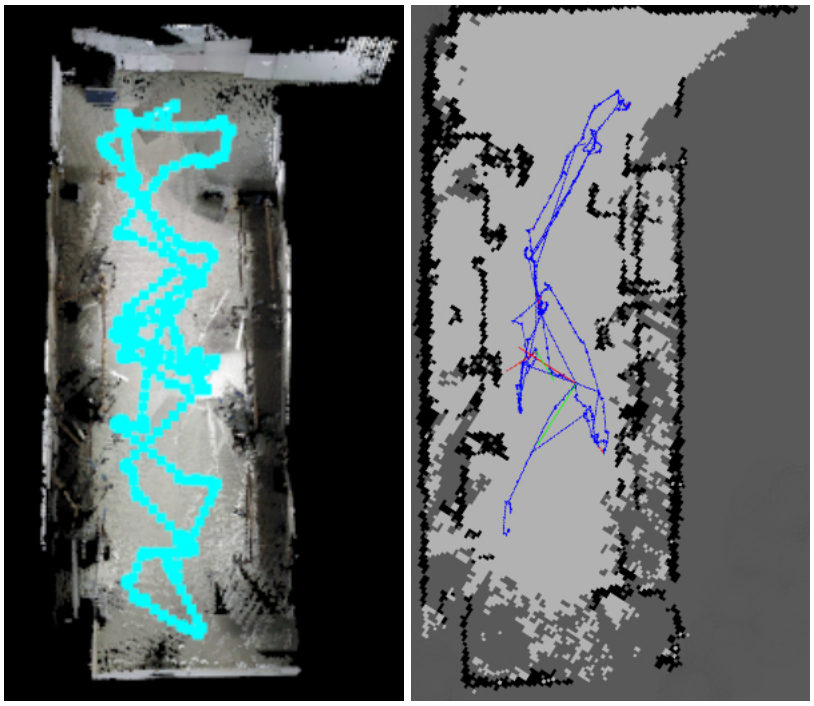

Figura 2.4: Esquerda OS-SLAM 3D e Direta OS-SLAM 2D

2.3 A Biblioteca de Nuvem de Pontos (PCL)

A descoberta da capacidade de novos *hardwares* de baixo custo no campo da visão computacional proporcionou à comunidade de pesquisa em robótica uma acessibilidade muito grande na área da percepção 3D [7]. A partir disso, a evolução nas pesquisas sobre a capacidade de um robô poder perceber o mundo externo deram um passo enorme. Anteriormente, eram usados sensores simples, como sonares ou sensores infra-vermelhos (IRs), os quais obtinham informações limitadas do ambiente e ainda tinham um custo bem elevado de mercado o que dificultava o acesso dos pesquisadores [11],[10].

Atualmente, surgiram os sensores 3D como Microsoft Kinect, câmeras RGB-D dentre outros, que além de terem um baixo custo tem a capacidade de fornecer nuvens de pontos 3D em tempo real, bem como imagens de profundidade utilizando-se de uma câmera gama e imagens em 2D. Isso tornou possível diversas pesquisas que apontam para a possibilidade dos robôs de perceberem o mundo em 3D de uma forma mais rica em detalhes [22],[29].

Sendo assim, esse baixo custo e essa capacidade de se poder manipular ou extrair do ambiente nuvens de pontos para uma melhor percepção do mesmo, foi que surgiu o trabalho de Radu Bogdan e Steve Cousins (2011) [31]. Eles apresentam o código aberto que denomi-

nam de Biblioteca de Nuvem de Pontos (do inglês *Point Cloud Library* - PCL), onde buscam criar uma aproximação da comunidade da robótica no tratamento de nuvens pontos e no desenvolvimento voltado para a percepção 3D, fornecendo um conjunto de aplicações voltadas a: segmentação , reconstrução, estimativa e filtragem dos dados obtidos através desses sensores capazes de obterem nuvens de pontos como matéria (ver Fig.2.5). Além de ser de código aberto e contar com pesquisadores colaborando para o desenvolvimento e evolução do projeto, a PCL ainda possui integração com o ROS (*Robotic Operation System*)[8] o que aproxima ainda mais o uso no campo da robótica [22][31].

Figura 2.5: Nuvem de Pontos Extraída através do Microsoft Kinect (Fonte:[22])

2.4 RGB-D SLAM

O uso dos sensores RGB-D e a capacidade da manipulação de nuvens de pontos proporcionou o surgimento de diversas novas abordagens de SLAM VISUAL. SLAM são algoritmos que dão aos robôs a capacidade de mapear e se localizar num ambiente anteriormente desconhecido, assim como a sigla refere-se SLAM(Mapeamento e Localização Simultâneos).

Uma delas chamada de RGB-D SLAM, capaz de reconstruir o ambiente 3D em que o robô está inserido de forma mais precisa e com um grau de aproximação e verdade muito alto. Tornando, assim, o mapeamento de áreas mais preciso e funcional no cenário da utilidade daquele mapa, proporcionando o surgimento de algoritmos de SLAM Visual com características diferentes, bem como a evolução do estado-da-arte [7],[11].

O RGB-D SLAM se caracteriza por ser a divisão no campo do SLAM, na qual as funcionalidades e possibilidades que os sensores RGB-D proporcionam foram exploradas em sua totalidade [12]. Foram descobertos, por exemplo, os avanços em relação à manipulação dos dados contidos nas nuvens de pontos para o mapeamento 3D, como também a possibilidade de uso das imagens para a identificação das informações de profundidades [33] [18]. Também descobriu-se a aplicação de algoritmos de análise de imagens para a manipulação desses dados nas abordagens de SLAM RGB-D como, por exemplo, o SURF [34] (ver aplicação do SURF no OS-SLAM na fig. 2.6) e RANSAC [35].

O algoritmo SURF (*Speeded up robust features*) [34] se caracteriza por ser um descritor de recursos locais e um detector de tarefas como, por exemplo, o reconhecimento de objetos, registro de características encontradas na nuvem de pontos, classificação e reconstrução 3D.

Já o RANSAC (*Random Sample Consensus*) [35] é um método de iteração capaz de estimar os parâmetros de um modelo matemático de forma aleatória. O uso desse algoritmo nos dados retirados das nuvens de pontos faz referência à identificação de planos nas nuvens captadas.

Assim, com as inúmeras possibilidades de uso desses dados obtidos das nuvens de pontos e sensores RGB-D, surgiram no cenário diversas abordagens. Cada uma delas apresentando diferentes tipos de tratamento ou extração de informações para a evolução do RGB-D SLAM e mapeamento 3D. Na Fig.2.7, observa-se um exemplo do SLAM RGB-D OS-SLAM de um ambiente de 7.55 m x 4.50 m [7] [19] [9].

Nas seções seguintes deste trabalho serão apresentados as mais recentes abordagens de SLAM RGB-D presentes no estado-da-arte e suas particularidades.

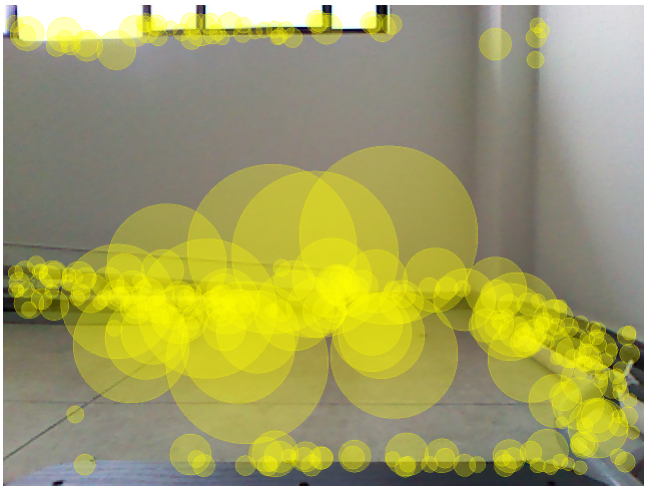

Figura 2.6: SURF Aplicado na Nuvem de Pontos do OS-SLAM Extraída através do Microsoft Kinect

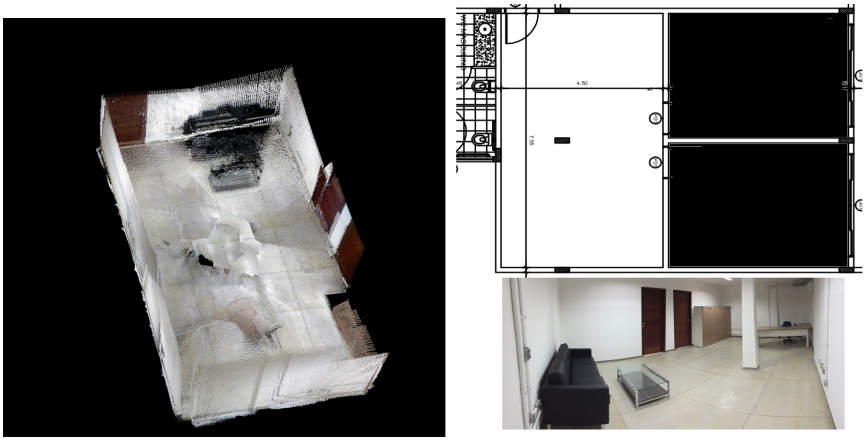

Figura 2.7: Esquerda OS-SLAM 3D e Direita Ambiente/Planta Baixa

2.5 Planar SLAM

Com as diversas possibilidades proporcionadas nos últimos tempos para o desenvolvimento da comunidade do RGB-D SLAM, tivemos o surgimento de algumas classificações dessas abordagens no estado-da-arte. Uma das mais atuais é o Planar SLAM [16], que será abordado neste tópico brevemente, buscando elucidar a classificação a que essa pesquisa está inserida e deixar claro o porquê dessa titulação.

A diferença do Planar SLAM para as demais abordagens do RGB-D SLAM, caracteriza-se por esses algoritmos atribuírem mais importância às características retiradas dos planos, encontrados nas nuvens de pontos ao decorrer do mapeamento [16], [17]. Nos algoritmos

de RGB-D SLAM convencionais, a nuvem é tratada de maneira que as operações de análise e depuração são aplicadas em todos os pontos em questão. Já no Planar SLAM, características provenientes dos planos encontrados serão a matéria e terão tratamentos e análises diferenciados [15], [18].

Estas abordagens defendem inúmeras vantagens que levam a utilização dessas características planares. Dentre elas, uma maior capacidade de percepção para os sistemas de SLAM 3D em grandes áreas, velocidade no mapeamento em tempo real, a possível capacidade de poupar o custo computacional em determinado cenário, dentre outras espalhadas nos demais trabalhos dessa área encontrados no estado da arte e que serão comentados em tópicos seguintes [10],[21],[19].

Capítulo 3

Trabalhos Relacionados

Neste terceiro capítulo é apresentado o estado-da-arte, na qual são compilados os estudos, mais recentes, relacionados ao trabalho em questão. Onde estão divididos em 2 sub-seções na primeira apresentaremos os avanços no campo do RGB-D SLAM e na segunda buscando afunilar ainda mais a classificação da pesquisa, são apresentados os trabalhos no campo do RGB-D Planar SLAM.

3.1 Avanços no RGB-D SLAM

Nesta seção, estão brevemente descritos os mais recentes avanços no campo de SLAM 3D quando utilizando o sensor RGB-D. A abordagem SLAM, usando sensores RGB-D, foi refinada como RGB-D SLAM [11][12]. O surgimento da câmera Kinect como um sensor RGB-D de baixo custo e sua capacidade de oferecer imagens em cores e de profundidade densas, levou à criação de uma nova abordagem que relaciona a informação de profundidade detectada com recursos visuais para criar uma representação densa de ambientes 3D.

O trabalho de N. Fioraio and K. Konolige [37] se caracteriza por ser um dos primeiros nas aplicações de Visual SLAM com as câmeras RGB-D, em que exploraram as capacidades que esses dispositivos baratos possuíam em relação à modelagem 3D, imagens de profundidade e imagens em cores, além da alta taxa de quadros que se era possível utilizar. Em seu trabalho foi proposta uma abordagem de mapeamento utilizando os dispositivos RGB-D, na qual fazem uso do algoritmo ICP alinhado com um *framework* de ajuste de feixe para combinar e comparar dois *frames* capturados num intervalo de 10 ms, fazendo o alinhamento global, em

tempo real, e criação do mapa 3D do ambiente.

O trabalho apresentado por Henry et. al. [33], propõe uma investigação da capacidade das câmeras RGB-D em construir mapas 3D do interior de ambientes, bem como executar uma análise destes mapas e suas aplicações no campo da robótica móvel. Além disso, é apresentada uma abordagem de mapeamento RGB-D, composto por um método de otimização aplicado no conjunto de formas obtidas através da extração de pontos das imagens RGB-D de entrada. O algoritmo faz uso do RANSAC para determinar as correspondências e do ICP para determinar o melhor alinhamento entre estes quadros. Os autores também demonstram a capacidade para detecção de fechamento de loop, que a abordagem também possui, através de um subconjunto de quadros coletados anteriormente, buscando, com isso, demonstrar que a otimização de gráficos pode ser conseguida através de alinhamentos globais consistentes.

Uma das abordagens pioneiras no campo do RGB-D SLAM é o trabalho de Endres et. al. [11], no qual foi apresentado um algoritmo de RGB-D SLAM *open source* que utiliza-se do Microsoft Kinect de mão para demonstrar a possibilidade da construir uma representação em 3D do ambiente e ,simultaneamente, calcular a trajetória realizada para tal. Os autores dividem o funcionamento desta abordagem em quatro etapas principais, em que primeiramente analisam as imagens RGB-D de entrada, buscando a identificação de características contidas nelas. Em um segundo momento, é feita uma associação dessas características contidas com as das imagens anteriores, denominadas agora de quadros-chaves. Já no terceiro passo, ocorre a avaliação das imagens de profundidade nos locais dos quadros chaves encontrados. E por fim, utilizando-se do RANSAC, os autores estimam as transformações relativas entres esses quadros. Este trabalho serviu como embasamento inicial para os avanços na utilização de câmeras RGB-D no campo do SLAM visual.

No trabalho realizado por Kerl et. al. [10] em 2013, os autores propõem um método de SLAM visual denso *open source* utilizando câmeras RGB-D, que minimiza tanto a foto métrica quanto o erro de profundidade sobre todos os pixels da cena. Além disso, eles propõem o que denominam de medida da similaridade, que se caracteriza por ser um método de identificação de fechamento de loop e seleção de quadros chaves ao decorrer do mapeamento. Buscando, assim, demonstrar que esses recursos contribuem para uma melhor exploração dos dados contidos nas imagens e numa melhor precisão na parte de estimativa de pose.

O trabalho apresentado por Hess et. al [7], traz uma abordagem de RGB-D SLAM capaz

de gerar mapas em 3D robustos que utilizam-se apenas de uma câmera RGB-D, o Microsoft Kinect, sem ter a necessidade de qualquer outro tipo de sensor/odometria. Buscando propor um sistema *open source* e de baixo custo para a comunidade, tem como diferencial sua aplicação nos cenários de pequenos robôs móveis, a exemplo de robôs aspiradores, quadcopteros, dentre outros. Não deixando, porém, de possibilitar o uso em mão livre, também, como os demais. Além disso, os autores propõem o uso de um modelo de mensuração de ambiente responsável por validar as estimações e transformações de correspondências realizadas pelo algoritmo de ponto mais próximo (ICP).

C. Kerl, J. Sturm e D. Cremers [36] , neste trabalho os autores demonstram que, além da capacidade de criação de mapas 3D, os dispositivos RGB-D ainda podem ser usados para criação das trajetórias realizadas por eles ao decorrer do mapeamento. Apresentando, assim, uma abordagem capaz de estimar o movimento realizado pela câmera RGB-D, através de minimização não linear e minimização de erro de fotométrica, além de propor uma função de erro que reduz a influência de outliers de grande porte. Demonstrando a possiblidade de estimativas para navegação do robô e controle de posição, calculando as atualizações de movimento em alta taxa de quadros e em tempo real. Todo o trabalho foi testado e desenvolvido num cenário voltado para robôs de capacidade computacional limitada, além de ter seu código disponibilizado à comunidade.

F. Steinbrücker et. al. [38], apresentam uma abordagem no mesmo campo da odometria visual e controle de posição através de dispositivos RGB-D baseada em energia. Através do uso do Microsoft Kinect, os autores propõem uma função de minimização de energia que tem como objetivo principal estimar o melhor movimento de um determinado corpo rígido através da associação de imagens RGB-D. Realizando uma linearização desta função para obtenção do que eles denominam de equação normal 6x6 voltada para as coordenadas de torção, onde são encontrados os movimentos sofridos pelo corpo rígido em questão. Esta abordagem propõe uma método que pode ser comparado no estado-da-arte ao uso do ICP.

Em 2014, a abordagem RTAB-Map foi proposta por Labb é e Michaud [9] . Em seu artigo foram abordados problemas que ocorrem nos algoritmos de SLAM para operações a longo prazo. Nestes algoritmos, um robô tem de lidar com o posicionamento inicial desconhecido causado por qualquer problema de robô sequestrado ou mapeamento multi-sessões. Os autores abordaram estes problemas, amarrando o sistema SLAM com uma abordagem de

detecção de fechamento de loop global, que intrinsecamente lida com estas situações. No entanto, o processamento em linha para o laço de detecção global de abordagens de fecho é geralmente influenciado pelo tamanho do ambiente. Por conseguinte, eles propõem a integração de um detector de fechamento de loop Bayesiano global proposto, pela primeira vez, pelos mesmos autores em 2011 [13] . A abordagem baseou-se num método de gestão de memória, o que limita o número de locais usados para a detecção de fechamento de loop. De modo que o tempo de cálculo se mantém sob restrições de tempo real. O detector de fechamento de loop utiliza uma abordagem saco-de-palavras para determinar a probabilidade de que uma nova imagem venha de um local anterior ou de um novo local. [14] . Esta abordagem foi capaz de cumprir os requisitos necessários para o processamento *on-line* em grande escala, de longo prazo e mapeamento *on-line* multi-sessão. Portanto, foi uma abordagem SLAM adequada para ser modificada em relação ao OS-SLAM, visando aumentar a precisão do mapa 3D.

3.2 Avanços no Planar SLAM

Planar SLAM é um tema recente na comunidade SLAM. Eles baseiam-se no pressuposto de que podem usar planos para construir um mapa do ambiente. O trabalho apresentado por Salas-Moreno et. al. [16] apresenta uma abordagem que mapeia densamente um ambiente, usando planos delimitados e características extraídas de imagens de profundidade, obtidas através das câmeras gama presente nos dispositivos RGB-D. Os autores afirmam que o ganho em relação ao mapeamento de grandes áreas torna-se mais eficiente, uma vez que o algoritmo apresenta um processo das regiões planas que permite representar mais precisamente suas medidas no mundo real.

Além disso, a abordagem apresentada no trabalho de Ma et. al. [17] explora a riqueza de informações fornecidas pelos métodos baseados no Planar RGB-D SLAM na área da robótica em tempo real. Os autores apresentam um método para triangulação eficiente e texturização de superfícies planas em grandes nuvens de pontos. O seu algoritmo de geração de textura preserva todas as informações de cor contidas em segmentos planares, resultando numa representação simplificada, visualmente atraente e geometricamente precisa.

Em 2012, Lee et. at. [15] apresentou um plano de extração de características rápido

e robusto e uma técnica de harmonização para sensores do tipo RGB-D. Eles propuseram três componentes necessários para utilizar os recursos do plano em um problema de linha no SLAM: extração de plano rápido, restrição de estimativa quadro-a-quadro, e planos que se fundem. Os autores associaram os recursos do plano com base em ambos os parâmetros da cobertura espacial, e estimaram as restrições estáveis pela função custo com prazo de regulamentação. O seu trabalho foi uma amostra do que seria a abordagem Planar RGB-D SLAM.

Taguchi et. al. [18] apresentam um algoritmo de Planar RGB-D SLAM voltado para sensores 3D de mão que tem como diferencial o uso dos pontos dos planos como primitivos. Realizando, asssim, a captação de dados 3D de maneira diferenciada, fazendo uso de 2 sistemas de coordenadas diferentes, onde qualquer combinação de 3 pontos primitivos serão úteis para o algoritmo como exemplos (3 planos, 2 planos e 1 ponto, 1 plano e 2 pontos, e 3 pontos). Os autores afirmam que com o uso de um conjunto de pontos de planos ao invés de representações baseadas apenas em pontos, resultam em informações de cena mais compactas e semânticas, bem como um ganho em tempo no que diz respeito à pesquisa de correspondência. Assim, quando ocorre um numero menor de planos comparado com a quantidade de pontos 3D, o algoritmo tende a escolher mais combinações envolvendo mais planos.

Além disso, Michael Kaess [19] propôs em seu trabalho uma fórmula que tem como principal função realizar uma representação mínima para características de planos, incluindo infinitos planos na formação de mapas. Essa abordagem faz uso de mínimos quadrados para otimização com Gauss-Newton, Powell Dog-Leg e ISAM. A abordagem tem como base, um mapeamento onde é realizada uma parametrização do plano homogêneo e a representação mínima para a otimização. Também é apresentada uma formulação relativa que melhora a convergência dos planos em questão.

R. Cabral e Y. Furukawa [20] demonstram um algoritmo para construção de modelos de planos 3D a partir de imagens. Os autores desenvolveram técnicas e demonstraram que utilizando–se da planaridade é possível criar estruturas 3D planas e em alta textura de um determinado ambiente tendo como dados brutos iniciais apenas uma imagem. No trabalho é proposto uma técnica para classificação de *pixels* determinando em que região ele deve se localizar, podendo ser piso, teto ou parede, sendo isso um dos pilares para reconstrução

em 3D se tornar possível a partir de imagens. Em um segundo estágio é demonstrado o uso das características dos planos na construção da estrutura 3D do ambiente (planta baixa), produzindo um modelo de planos e controlando a densidade da nuvem de saída.

Em contraste, Xiang Gao e Tao Zhang [21] também apresentam em seu trabalho uma nova abordagem de RGB-D Planar RGB-D SLAM, onde propõe um método de detecção que faz a extração e seleção de características de profundidade de confiança, que denominam de características planares ou dos planos. Segundo os autores, tais características contêm vantagens de exatidão e solidez quando trabalhado com o algoritmo de ponto mais próximo (ICP). Essas vantagens reduzem o erro de estimativa presente no alinhamento de características de pontos para determinação de posição, que é calculada pelo ICP através das informações de profundidade obtidas dos sensores. Outra qualidade proveniente do uso das características dos planos é demonstrada pelos autores através da conservação do custo computacional para a utilização nas aplicações em tempo real.

Capítulo 4

SLAM Baseado em Subtração de Objetos

Neste capítulo é apresentada de maneira detalhada toda a estrutura da abordagem OS-SLAM desenvolvida neste trabalho. Contendo seções que esmiunçarão cada etapa do funcionamento do algoritmo, bem como pseudo-códigos, diagramas, figuras e equações que elucidarão ainda mais o entendimento do trabalho.

4.1 Formalização do Problema

Ser capaz de construir um mapa do ambiente e localizar-se, simultaneamente, dentro deste mapa é uma característica comum em sistemas robóticos, ao navegar em ambientes desconhecidos, na ausência de informações anteriores. Uma abordagem baseada em RGB-D SLAM constrói um problema de estimação simplificado, abstraindo as medições brutas do sensor. Estas medições brutas são substituídas pelas margens do gráfico que podem ser visto como "medições virtuais". Em contraste, a abordagem planar SLAM baseia-se no pressuposto de que se pode utilizar as características dos planos para construir um mapa do meio. No entanto, as abordagens anteriores sofrem à influência de objetos móveis, quando inseridos no ambiente. Embora os objetos móveis sejam estáticos, qualquer alteração na cena requer um novo mapeamento do ambiente.

Como acima mencionado, o RTAB-Map tem mostrado bons resultados no mapeamento e localização (SLAM) de sistemas robóticos em ambientes grandes e complexos. Em contraponto, o mapa gerado ainda apresenta objetos integrados nos seus resultados [9]. Este trabalho propõe uma melhoria ao algoritmo RTAB-Map e o método resultante é chamado

de SLAM baseado em Subtração de Objetos (OS-SLAM). A abordagem OS-SLAM é um algoritmo baseado em Planar RGB-D SLAM que tem o seu objetivo distintivo para remover objetos (exceto paredes ou pisos) do mapa final. A contribuição desse trabalho é a heurística que os maiores planos no ambiente são as paredes e/ou chão, e todos os outros objetos são apenas móveis/pessoas. A heurística é implementada através da execução do algoritmo RANSAC para identificação de planos na nuvem de pontos de entrada. Os pontos que não correspondem a qualquer dos planos encontrados são "objetos". O sistema, então, trata os objetos e os projeta no plano mais próximo, subtraindo-os/achatando-os do ambiente (buscando uma melhor visualização dos resultados as imagens dos objetos ficaram espelhadas nos planos a que foram removidos). Melhorando, portanto, a localização, bem como aumentando a precisão do mapa gerado.

A existência de objetos móveis e/ou pessoas no ambiente torna difícil a criação de um mapa genérico e preciso. Uma vez que o mapeamento ocorreu com os objetos e pessoas, esses irão ser elementos que podem desaparecer ou mudar de posição a qualquer momento, tornando-se necessário refazer todo o mapeamento. A capacidade que o OS-SLAM tem de remover objetos/pessoas durante os resultados do mapeamento resultará em um mapa contendo apenas estruturas fixas - como paredes e/ou chão - ou seja, um mapa definitivo.

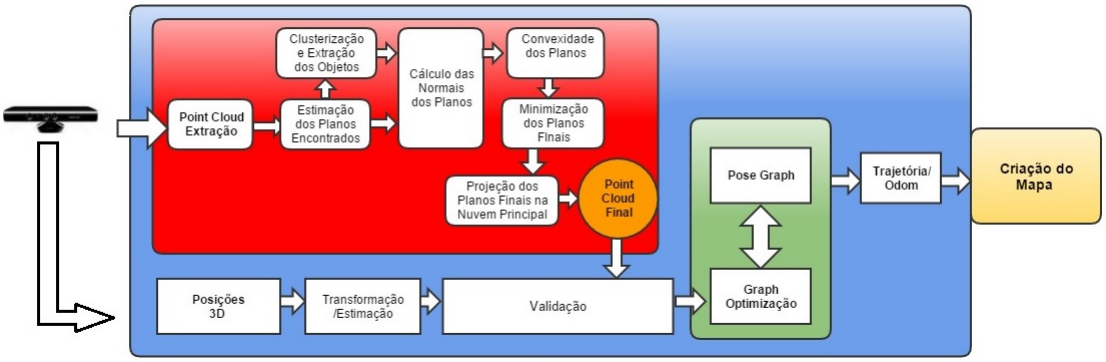

Figura 4.1: Diagrama da Estrutura do Algoritmo

4.2 O Algoritmo Proposto

O que diferencia o OS-SLAM de outras abordagens de SLAM 3D [15] diz respeito à detecção de objetos durante a etapa de mapeamento. Fig.4.1 apresenta uma visão esquemática

da nossa abordagem. As modificações introduzidas no algoritmo de mapeamento RTAB são representadas na área vermelha que mostra o ciclo que é aplicado para a nuvem de pontos de entrada e, consequentemente, para a nuvem final que vai ser o mapa visual. O ponto de turvação do ambiente é captado pelo sensor RGB-D e, em seguida, move-se para a segunda fase, onde é aplicada a estimativa de planos. Após o agrupamento dos objetos detectados, ocorre o processo de clusterização dos mesmos para que possam ser removidos da nuvem durante o terceiro passo descrito na zona vermelha, resultando em um mapa mais claro no que se refere à intervenção de objetos dinâmicos ou móveis.

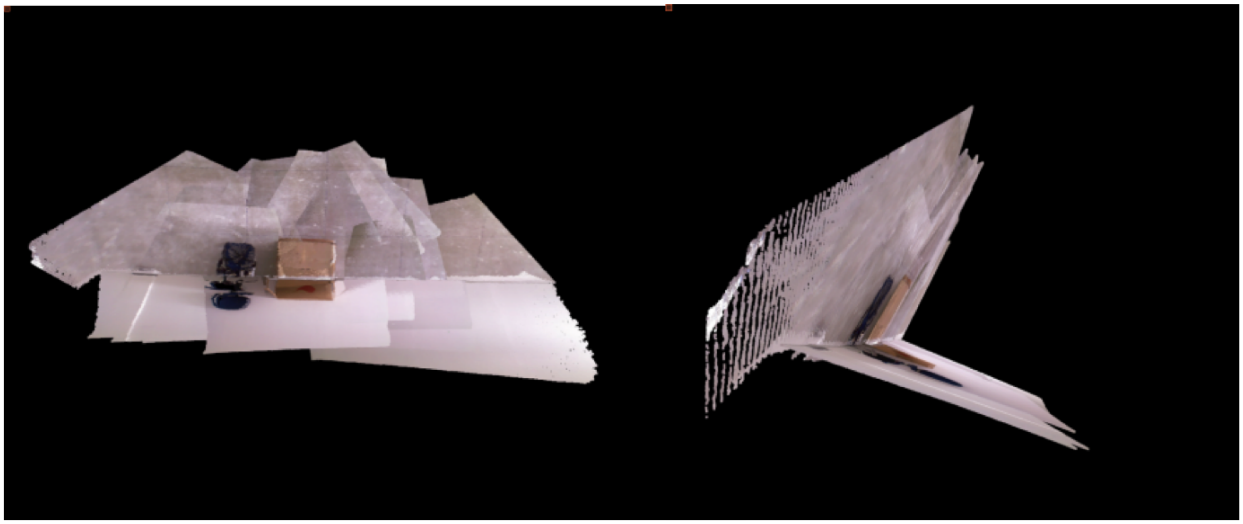

Figura 4.2: Efeito do Algoritmo OS-SLAM

A maior vantagem de usar a identificação de objetos no mapeamento em um processo de RGB-D Planar SLAM é a possibilidade de gerar mapas mais genéricos e precisos. De alguma forma, os objetos em cena podem interferir na criação de mapas em 3D, sendo considerados como parte integrante desta, gerando distorções no mapa. Ao excluir esses objetos, o algoritmo OS-SLAM pode gerar mapas que irão conter apenas solo e parede, dando uma percepção melhor do espaço sem mobília, mesmo em casos como mesas (ver Fig 4.2). Nos experimentos executados as únicas exceções foram, por exemplo, estantes ou armários, pois para tal usamos a plataforma Turtlebot 2 (ver Fig. 2.2). De modo que, a partir de sua perspectiva e percepção, esses objetos são muito altos e foram confundidos com paredes.

Usando a *Point Cloud Library* (PCL) [22], as melhorias propostas neste trabalho são realizadas na classe de aquisição da nuvem crua do RTAB-Map, e no módulo onde os ponto-

chaves serão combinados através da extração de características pelo SURF. O SURF geralmente detecta mais pontos chaves para que os *inliers* também sejam grandes, enquanto SIFT, ESTRELA têm menos pontos-chaves que podem levar a problemas inexpressivos [21] . Os principais passos do OS-SLAM são enumerados como segue:

1. Estimação dos planos (paredes e piso) via RANSAC;
2. Identificação Clusterização e detecção dos objetos;
3. Estimação Final dos Planos:
   (a) Extração da convexidade e Estimação da normal dos Planos;
   (b) Minimização e Planos finais Estimados;
4. Projeção dos Planos Finais + Os objetos identificados e removidos na Nuvem.

4.2.1 Estimação e Remoção dos planos relativos a Paredes e Piso

O primeiro passo do OS-SLAM é obter a matéria que é a Nuvem de Pontos do sensor RGB-D e estimar o modelo dominante de planos através do algoritmo RANSAC. Os modelos de planos retornados nesta abordagem são principalmente paredes e pisos, esses serão utilizados durante a etapa de segmentação.

O modelo de plano pode ser representado de uma forma de parametrização esférica $\psi = (\alpha, \beta, d)$ [15]. Dado o $\bar{p}$ significativo do plano, N como o vetor da normal do plano, e (n) como uma função que transforma um vetor normal em ângulos no sistema de coordenadas esférica. Os parâmetros de plano pode ser obtido na forma de:

$$\psi = \begin{bmatrix} \alpha \\ \beta \\ d \end{bmatrix} = \begin{bmatrix} \Theta(n) \\ \bar{p}^T(n) \end{bmatrix} \tag{4.1}$$

Suponha que C_p é a matriz de covariância dos pontos do plano, e tem valores próprios ($\lambda 0$, $\lambda 1$, $\lambda 2$) e autovetores ortonormais (e_0, e_1, e_2), onde $\lambda 0 < \lambda_1 < \lambda_2$ e $e_0 \times e_1 = e_2$. O plano de coordenadas quadro pode ser definida de tal forma que: a origem é de US $\bar{p}$ e (x_p, y_p, z_p) eixos correspondem a (e_0, e_1, e_2) . Parâmetros do plano tornar-se:

$$(\alpha_p, \beta_p, d_p) = (0, 0, 0); \quad \bar{p}_p = (0, 0, 0). \tag{4.2}$$

Os parâmetros do plano podem ser transformados dado uma transformação T $= (R, t)$, entre os dois quadros da câmara, como se segue:

$$\psi' = \begin{bmatrix} \alpha' \\ \beta' \\ d' \end{bmatrix} = \begin{bmatrix} \Theta(RN(\alpha, \beta)) \\ t^T RN(\alpha, \beta) + d \end{bmatrix} \tag{4.3}$$

$$\bar{p}' = R\bar{p} + t \tag{4.4}$$

onde N $(, \beta)$ transforma os ângulos a um vetor da normal a ser aplicada na rotação.

O erro covariante dos parâmetros do plano $C_p\psi$ e de ponto de distribuição C_p pode ser derivado da seguinte forma:

$$C_{\psi'} = J_{T,\psi} \begin{bmatrix} C_T & 0 \\ 0 & C_\psi \end{bmatrix} J_{T,\psi}^T \tag{4.5}$$

$$C_{p'} = J_{T,p} \begin{bmatrix} C_T & 0 \\ 0 & C_p \end{bmatrix} J_{T,p}^T \tag{4.6}$$

onde C_T é a matriz de covariância da transformação T, $J_{T,\psi}$ e $J_{T,p}$ representam a matriz Jacobiana da equação 4.5 e 4.6, respectivamente.

Esses planos detectados são removidos temporariamente para clusterização dos objetos e depois reinserido na etapa 3.

4.2.2 Identificação e Clusterização dos Objetos Detectados

O próximo grande passo no *pipeline* de processamento é a extração de objetos. Os objetos candidatos são todos os pontos não relacionados com grandes planos, por exemplo: cadeiras,

caixas, mesas, etc. Para este segmento no conjunto de objetos foi necessário identificar aqueles que encontram-se sobre ou na frente do plano principal. Tal como referido antes, os modelos de planos são extraídos a partir do ponto de turvação original, deixando todos os outros pontos não relacionados com planos, neste caso os objetos. Esta etapa termina com a aplicação de uma segmentação usando o método de clusterização euclidiana. O algoritmo de extração de objetos pode ser visto no Algoritmo 1.

Algoritmo: Pseudo-Código de Extração e Clusterização dos Objetos

ENTRADA (Nuvem com Objetos) **– Nuvem apenas com os Objetos**;
SAÍDA: (Segunda Nuvem sem Planos (Nuvem Clusterizada));
1: *Get* (Nuvem Objetos) *points*; *{Nuvem sem Planos}*
2: Clusterização dos Objetos usando a (*PCL Euclidean Cluster Extraction class)*
3: Setando entrada da Nuvem (Nuvem Objetos);
4: Extração dos Indices (*Objectindices*);
5: **for**($it = Objectindices.begin : Objectindices.end$)**do**
6: **for**($pit = it >> indices.begin : it >> indices.end$)**do**
7: cloud>>points[*pit].r = 255;
8: cloud>>points[*pit].g = 255;
9: cloud>>points[*pit].b = 255;
10: **end for**
11: **Passo de SAÍDA** (Cloud com Objetos Identificados e Clusterizados);
12: **end for**
13: **return** Nuvem Clusterizada;

A extração de cluster euclidiana retorna um vetor de índice de pontos dos objetos. Esses índices serão utilizados para selecionar todos os pontos relacionados aos objetos, a partir da nuvem de pontos original. Depois disso, o usuário pode definir uma cor para o preenchimento destes pontos, em prol de fazê-los parecer tão quanto os planos estimados em torno deles. Neste trabalho não foram removidos os objetos para melhor perceber o processo de exclusão, considerando como parte do terreno e/ou de primeiro plano.

4.2.3 Estimação do Planos Finais

Neste passo, tratou-se os planos extraídos a partir da nuvem principal. Este processo foi dividido em duas etapas subsequentes. Na primeira etapa, calcula-se a normal e convexidade desses planos. Em seguida, aplica-se um plano de minimização baseado no algoritmo de minimização norma l_0 [23] .

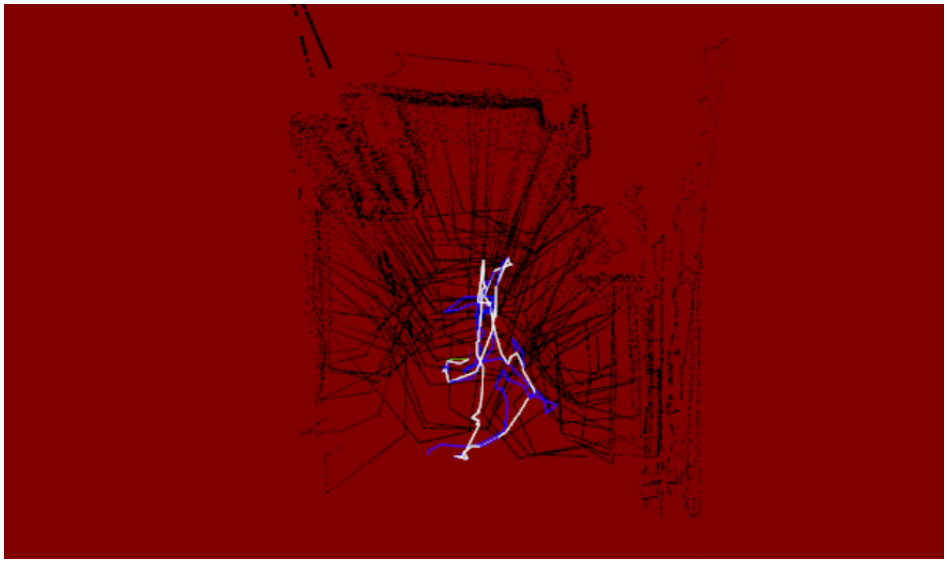

Figura 4.3: Nuvem apenas de Planos

4.2.4 Estimação da Normal e Extração da Convexidade dos Planos

Vários fatores relacionados ao tempo de filmagem de cada cena podem causar planos inacabados ou planos paralelos na nuvem. Entre estes fatores tem-se a influência da luz, sombra de objetos e mesmo a distância a partir do sensor. Portanto, as três etapas da estimativa dos planos finais descritos aqui (normal, convexidade e aplicação da norma de minimização l_0) resultam em uma nuvem final com planos mais precisos e definidos.

Fig. 4.3 apresenta uma visualização da nuvem apenas com planos elaborados a partir da nuvem principal. Essa nuvem será o assunto das próximas três etapas neste tópico em que vamos tratar as duas primeiras sub-etapas através da estimativa da normal da eq. 4.1 e calcular a convexidade usando Algoritmo 2.

Algorithm 2: Pseudo-Código da Estimação das Normais/Extração da Convexidade

ENTRADA (Nuvem de Planos + Nuvem Clusterizada);

SAÍDA: (Nuvem contendo apenas planos (Depois do Processo das Normais e Convexidades));

1: Pontos de Estimação da Normal (Nuvem de Planos); *{Nuvem apenas com Planos}*

2: Segmentação das Normais usando a (*PointT class*);

3: *NormalsWriter*(Nuvem de Planos (N)); *{Normais dos planos}*

4: Extração dos Indices usando a classe (*PointT class* - Nuvem de Planos/Convexidade (C));

5: Inclusão dos Indices dos Objetos (Nuvem de Planos (N) + (C));

6: Extração dos Melhores Planos (Nuvem de Planos (N) + (C));

7: Projeção na Nuvem (Nuvem de Planos (N) + (C)) usando *Project Inliers class*;

8: Inclusão das Nuvens (*CloudOut* = Nuvem de Planos + Nuvem Clusterizada);

9: Minimização dos Coeficientes dos Planos (*CloudOut*);

10: ***ETAPA DE SAÍDA*** (Nuvem de Planos + Nuvem Clusterizada (*CloudOUT*));

11: **return** *CloudOut*;

4.2.5 Minimização dos Planos Finais

Antes da etapa final desta abordagem (isto é, a projeção de todos os planos estimados), tem-se de eleger os planos mais significativos. Portanto, fora incluído o algoritmo de minimização norma l_0 [23] para priorizar planos que representam paredes e pisos. Esta etapa minimiza os planos menores e paralelos que são criados durante a captura da nuvem de pontos. Resultando, assim, num mapa que contém planos referentes apenas a piso/parede 3D (ao final do processo e depois da remoção de objetos móveis) que é mais preciso, e melhor visualizado com um maior nível de verdade. Abaixo, é mostrado o algoritmo de minimização norma l_0 que foi alterado para aplicação no sistema OS-SLAM aqui abordado. Usamos o algoritmo norma l_0 para ser possível eleger os planos mais significativos achados nas etapas anteriores. Para isso, as modificações feitas nele incluem a capacidade de atribuir pesos aos planos tomando em consideração valores como normal do plano e convexidade. Sendo possíve, assim, aplicar a minização e exclusão de planos paralelos ou pequenos demais oriundos de alguma sombra ou de algo do tipo.

Para apresentar o algoritmo, primeiro temos de seguir a definição da norma l_0 de um vetor $\mathbf{s} = [s_1, \cdots, s_n]$ como o número de componentes não nulos de $\mathbf{s}$. Para calcular isso, primeiro definiu-se:

$$\nu(s) = \{\, 1\,, s \neq 00, s = 0 \tag{4.7}$$

Então a norma l_0 de s pode ser escrita como:

$$\|\mathbf{s}\|_0 = \sum_{i=1}^{N} \nu(s_i) \tag{4.8}$$

É claro que esta norma é descontínua, desde que Nu é uma função descontínua. Contornou-se isso empregando uma aproximação suave de US $nu(s)$. Muitas funções podem ser usadas, e foi escolhido uma função Gaussiana de média nula devido à sua diferenciabilidade. Ao definir:

$$f_\sigma(s) = exp\left(\frac{-s^2}{2\sigma^2}\right), \tag{4.9}$$

Nos temos:

$$\lim_{\sigma \to 0} f_\sigma(s) = \{\, 0\,, s \neq 01, s = 0 \tag{4.10}$$

Consequentemente, $\lim_{\sigma \to 0} f_\sigma(s) = 1\nu(s)$, e, portanto, define-se:

$$F_\sigma(\mathbf{s}) = \sum_{i=1}^{N} f_\sigma(s_i), \tag{4.11}$$

Temos:

$$\lim_{\sigma \to 0} F_\sigma(\mathbf{s}) = N - \|\mathbf{s}\|_0. \tag{4.12}$$

Por isso toma-se em seguida, $N - F_\sigma(\mathbf{s})$ como uma aproximação $\|\mathbf{s}\|_0$:

$$\|\mathbf{s}\|_0 \approx N - F_\sigma(\mathbf{s}), \tag{4.13}$$

onde σ é o trade-off entre a precisão e a suavidade da aproximação.

A partir da equação 4.13, a minimização de $\|Z\,|_0$ norma é equivalente à maximização de $F_\sigma(\mathbf{s})$ para um σ suficientemente pequeno.

Para notação da simplicidade, permite reescrever a equação problema convexo como x = A B *, onde B é o vector de ponderação e A é a matriz de medições retardada. É preciso, também, como ponto de partida, da pseudo-inversa da matriz a, que denotamos quanto A, e a partir do qual se encontra o primeiro palpite para B na forma de $B = A$.

Tomou-se o palpite e multiplicou-se ele por uma fração ponderada de uma aproximação de seus US norma l_0, para cada célula do vetor. Aqui, a aproximação é definida pelo desvio padrão de uma função gaussiana de média nula, chamado aqui simplesmente como a Sigma.

Então, foi subtraída B a residual encontrada projetando o erro sobre a matriz pseudo-inversa. Depois de repetí-lo um número satisfatório de vezes, diminuiu-se o valor do sigma. Todo este processo foi repetido até que o sigma atingiu o sigma de referência, que foi o *proxy* para a precisão. O pseudo-código para isso pode ser visto no Algoritmo 3.

Algoritmo 3: Norma de Minização l_0

ENTRADA B=SL0(A,x,sigma_min,A_pinv);

SAÍDA: *Weight vector* B *that solves* $x = A * B$;

1: Inicialização;
2: $B = A\_pinv * x$;
3: $sigma = 2 * max(abs(B))$;
4: $mu_0 = 2$;
5: $L = 3$;
6: $sigma\_decrease\_factor = 0.5$;
7: **while** $(sigma < sigma\_min)$ **do**
8: **for**$(i = 1 : L)$ **do**
9: $delta = B. * exp(-abs(B).^2/sigma^2)$;
10: $B = B - mu_0 * delta$;
11: $B = B - A\_pinv * (A * B - x)$;
12: **end for**
13: $sigma = sigma * sigma\_decrease\_factor$;
14: **end for**
15: **return** $sigma$;

4.2.6 Projeção dos Planos Finais

A etapa final da abordagem consiste em projetar todos os objetos e pontos no modelo de plano estimado. Como resultado, é possível obter uma nuvem contendo apenas os pontos de planos estimados e os pontos de objetos achatados em tais planos.

4.2.7 OS-SLAM Multi-Robô e Merging

Nos parágrafos anteriores, o OS-SLAM foi explicado no que diz respeito à sua capacidade para a criação de mapas gerais usando sua característica de identificar e subtrair objetos móveis existentes no ambiente de mapeamento, priorizando a captura de paredes e piso. Esta abordagem também fornece uma alternativa para a utilização do algoritmo num cenário de mapeamento com múltiplos robôs. Como mostrado acima, o OS-SLAM realiza um tratamento nas nuvens de pontos capturadas pelo Kinect, criando o mapa final dessa área. Depois de todo o processo de mapeamento, o algoritmo resulta em um banco de dados contendo o mapa e a odometria.

Em um cenário de multi-robot, cada robô irá executar o algoritmo de forma independente e será responsável pelo mapeamento de uma determinada área do ambiente em questão (ver Fig.4.4). Isto é, o ambiente vai ser dividido e cada robô será responsável por mapear uma área. Para o algoritmo ter a capacidade de mesclar os mapas individuais apresentados por cada robô, é necessário que todos eles tenham revisitado pelo menos uma área comum. A área revisitada irá fornecer pontos-chave em comum entre as nuvens geradas individualmente por cada robô. No servidor será aplicado o RANSAC quantificado utilizando as correspondências pelo algoritmo SURF entre as nuvens para criar a ligação entre os mapas e conseguir a geração do mapa global. A partir disso, será possível identificar onde cada mapa, feito individualmente, se encaixará, sendo apto a gerar um único banco de dados que contém o mapa global do ambiente.

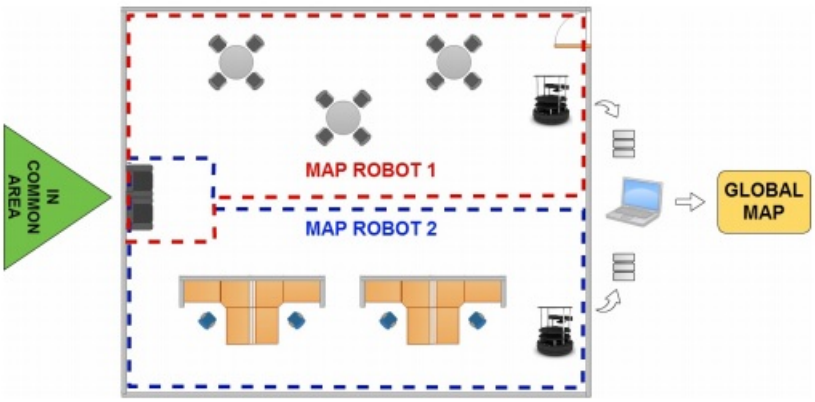

Figura 4.4: Diagrama Merging

Capítulo 5

Avaliação Experimental

Nesta seção, foram realizados quatro experimentos para demonstrar a eficiência da abordagem. Em três deles foram utilizados cenários reais, como o Laboratório de Sistema Embarcados e Robótica (LaSER), bem como o terceiro andar do Centro de Informática da UFPB. O experimento restante foi feito através da aplicação do *Groundtruth* [26] para validação quantitativa do OS-SLAM.

5.1 Configuração dos Experimentos

Os resultados foram feitos utilizando a plataforma Turtlebot 2 com um Kinect Sensor versão 1. Ambos, Turtlebot e sensor Kinect, foram conectados em um Notebook Lenovo rodando o ROS e Ubuntu 12.04: 1,86 GHz CPU Intel Core i5, 500 GB de HD e 4 GB de RAM. Nenhum outro tipo de sensor foi usado nestes experimentos. Os parâmetros OS-SLAM podem ser vistos na Tabela 5.1.

Alguns valores sobre as definições de odometria tiveram de ser alterados, a fim de obter um melhor desempenho. Em primeiro lugar, a distância máxima para correspondências 3D foi alterada de 0.020m para 0.050m, reduzindo a precisão, mas permitindo um melhor desempenho. Este parâmetro aumenta a área de pesquisa a partir de um quadro para outro, quando correspondendo os pontos de características extraídos. Também aumentou a visão de profundidade para correspondências (profundidade máxima de recurso) até 5m. O limite de distância de estimativa de planos RANSAC foi criado para $0,01m$, enquanto a tolerância de cluster foi definida para $0.01m$. Isto significa que todos os pontos que são 1 equidistante

Tabela 5.1: OS-SLAM Parâmetros

Parâmetros	**OS-SLAM Valores**
Distância Máxima para as correspondências 3D	0.050m
Profundidade Máxima	5m
Distância Limiar	0.01m
Tolerância de Cluster	0.01m
Min Tamanho de Cluster	100 points
Max Tamanho de Cluster	25000

fazem parte do mesmo grupo. O tamanho min cluster e tamanho do cluster max define o número mínimo e máximo de pontos a serem considerados em um cluster. Finalmente, um vídeo foi criado para explicar melhor os dois primeiros experimentos discutidos abaixo.

5.1.1 TurtleBot OS-SLAM 3D do Laboratório

Esta primeira experiência foi realizada sobre o ambiente apresentado na Fig. 5.1 utilizando um robô Turtlebot 2. Este ambiente é do laboratório de Sistemas Embarcados e Robótica (LaSER). A sala é 9,65m x 3,70m de tamanho e ela é preenchida com um conjunto diversificado de objetos com diferentes tamanhos e formas, como cadeiras, computadores, mesas, robôs, lata de lixo, etc.

Este experimento teve o objetivo de avaliar a precisão na geração de mapas através da etapa de mapeamento do OS-SLAM e foi utilizado o trabalho [14] para comparação. Fig.5.2 mostra que os objetos dispostos em vários locais do laboratório foram destacados (canto superior esquerdo). Algoritmos de mapeamento convencionais são fortemente afetados pelos objetos junto às paredes. Esses objetos criam oclusões e podem interferir na construção de mapas.

Tal como referido antes, RTAB-Map é um algoritmo de RGB-D SLAM eficiente. Pode mapear o ambiente apenas com o sensor RGB-D e de odometria também. No entanto, neste primeiro experimento, além do mapeamento o (OS-SLAM) fez também a detecção de objetos estáticos (móveis) durante a criação do mapa e os removeu do mapa suprimindo–os em planos criados para aumentar a precisão na construção do mesmo. O objetivo era recuperar

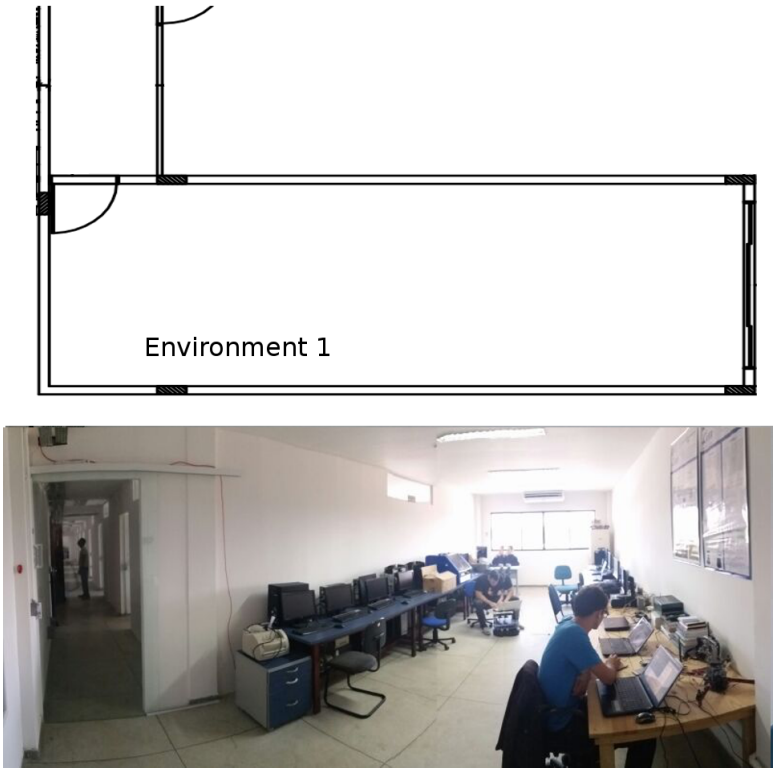

Figura 5.1: Ambiente do Laboratório Mapeado

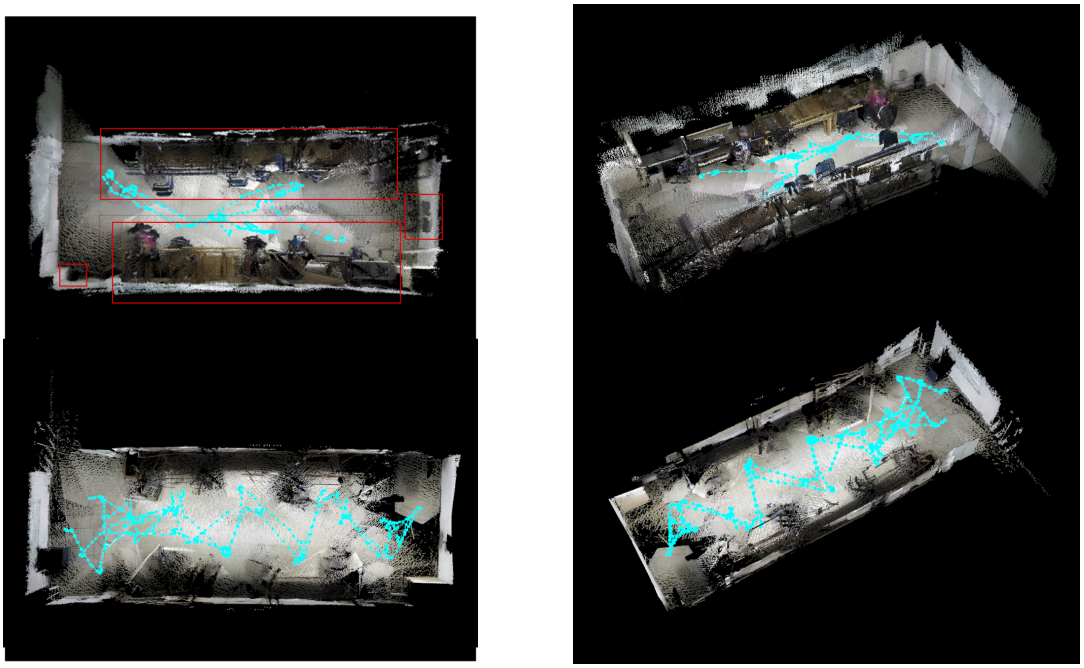

Figura 5.2: Vista Superior do RTAB-Map (topo esquerda) e OS-SLAM (abaixo esquerda). Vista Diagonal do RTAB-Map (topo direta) e OS-SLAM (abaixo direita)

o mapa original sem objetos para comparar os resultados obtidos nos ensaios com os objetos (RTAB-MAP). Nota na Fig.5.2 (canto superior esquerdo) que o mapa gerado utilizando RTAB-Map cria distorções nas paredes, enquanto em OS-SLAM (canto inferior esquerdo) as paredes são quase perfeitamente em linha reta. Estas correções são realizadas com a aplicação do tratamento nos planos e nos objetos detectados resultando em paredes mais planas. Pequenos valores atípicos podem ser criados durante o processo de mapeamento com o algoritmo RTAB-Map. No RTAB-Map, os valores atípicos são apresentados tanto em paredes longas como em cruzamentos de parede. Quando foi aplicado o algoritmo OS-SLAM apenas em junções de parede *outliers* foram apresentados. Essa diferença pode ser explicada pelo fato de que o algoritmo tem o foco sobre os cálculos das caracteristicas dos planos, evitando a criação de planos superpostos e suprimindo-os objetos encontrados em vez de mapeá-los como parte do fundo.

Fig.5.2 também mostra a visão diagonal dos mapas 3D gerados (topo e figuras de fundo à direita). Esses números apresentam o mapa 3D com a trajetória realizada pelo robô Turtlebot 2. Nota-se que no caso do RTAB-Map as paredes, por vezes, não foram bem detectadas, e ocorreram a criação de pequenos planos paralelos. Este problema de montagem de plano é causado pelo alcance do Kinect V1. A resolução e precisão deste dispositivo não é linear e a equação que converte o "quantum"de metros pode ter erros relevantes (mais de 3 cm) em alguns casos. É por isso que abordagens probabilísticas são usadas frequentemente. Além disso, são computacionalmente ineficiente para pequenas aplicações robóticas (requisitos de tempo real e restrições de energia). No entanto, na nossa abordagem esses erros não ocorrem devido à minimização $norma$ l-0, convexidade e cálculos das normais. Na maioria dos casos [9] , autores usam varreduras a laser além do sensor Kinect para aumentar a precisão na montagem dos planos. Além disso, o objetivo principal foi também alcançado: a remoção de objetos dentro do ambiente.

Finalmente, Fig 5.3 apresenta gráficos e na tabela 5.2 temos de forma mais visível os dados, a fim de quantificar o poder de processamento do OS-SLAM em relação ao tempo (ms) no experimento descrito acima. Foram escolhidas as informações mais relevantes para demonstrar o desempenho do algoritmo. No gráfico 1 (Nos indexados/tempo de processamento - superior esquerdo) tem-se as diretrizes para entender os gráficos seguintes. A eixo **X** mostra o número total de nós contidos no mapa global, enquanto o eixo **Y** apresenta o tempo

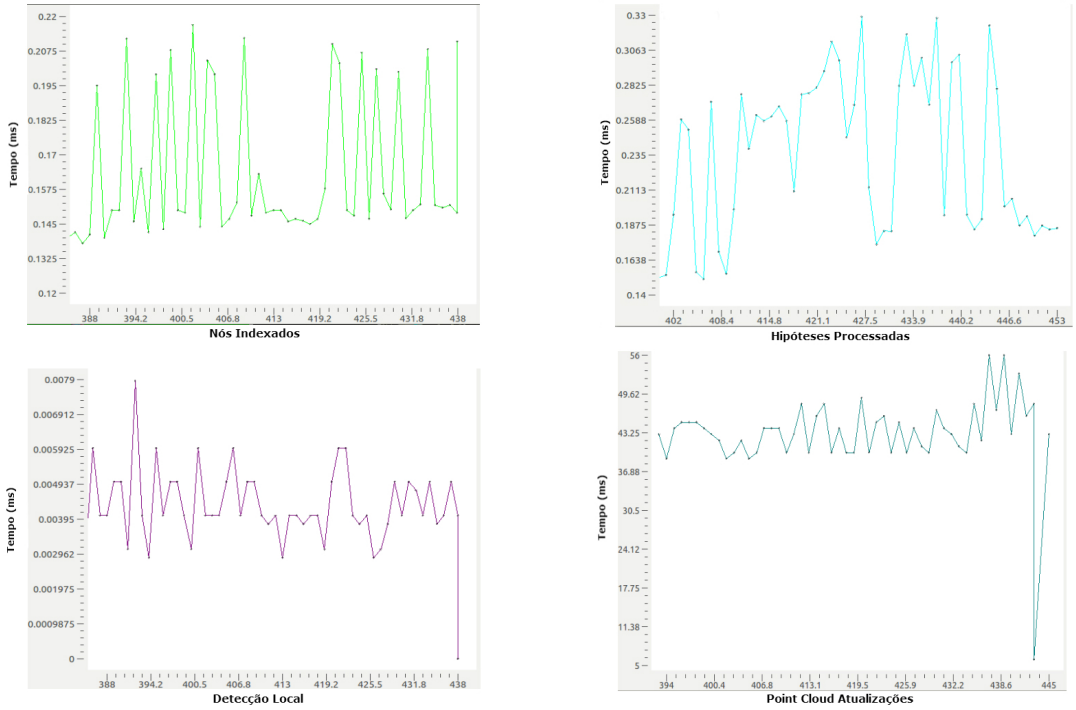

Figura 5.3: Estatistísticas de tempo do OS-SLAM no Experimento do Laboratório

Tabela 5.2: Detalhamento dos Resultadoss Obtidos no Experimento do Lab

Parâmetros	**Detecção Local**	**Nós Indexados**	**Hipóteses Processadas**	**Pc Atualizações**
Nós	438	438	438	438
Máx (ms)	0.0079	0.22	0.33	56
Min (ms)	0.0039	0.135	0.15	5
Mediana (ms)	0.00395	0.167	0.235	30.5

em milissegundos que era necessário para a identificação e processamento de cada nó no momento do mapeamento. Neste gráfico, vê-se que o algoritmo levou 0,22 ms para indexar um nó na nuvem de pontos, e adquiriu um total de 438 nós neste experimento. No gráfico *Hipóteses Processado* da figura 5.3. (Canto superior direito), mostra-se o tempo necessário (0,33 ms) para o processamento da aplicação da análise, extração de objetos e tratamento de planos nos nós adquiridos (total de 438). A diferença de tempo entre este e o grfico anterior é justificada pelo fato de que o Gráfico 3 (*Detecção local* - canto inferior esquerdo) mostra apenas o tempo (0,0079 ms) necessário para a captura de uma cena em particular, durante o mapeamento em tempo real. O processo de detecção de locais tendem a ser mais rápidos

do que o processo de indexação de nós, uma vez que a detecção de cena é a fase antes dos nós serem indexados, como ilustrado na Fig.4.1. No Gráfico 4 (*Nuvem Ponto Atualizações* - parte inferior direita) é visível o grande aumento no tempo em comparação com gráficos anteriores, uma vez que tem-se um tempo de 54.6 ms para o algoritmo realizar todo o seu processamento e incluir uma determinada parte do mapa na nuvem final.

A primeira parte do vídeo por Nascimento et. al. [24] apresenta o SLAM do mapeamento do Laboratório LaSER realizado pelo Turtlebot. Quando a experiência é iniciada, o robô mapeia o Laboratório suprimindo os objetos ao se mover. O robô inicia de frente para uma mesa e uma cadeira, movendo-se, então, para uma porta de vidro. Como ele se vira e encara uma gaveta do escritório pequeno, ele também suprime. Durante o experimento, o robô vai perto das mesas para melhor visualizar as paredes por trás deles. É notável que nem mesmo as mesas foram um problema para a abordagem proposta. A única exceção foi a estante de livros que, provavelmente devido a sua altura, o robô visualizou como uma parede.

5.1.2 TurtleBot OS-SLAM 3D de um Ambiente Grande

O segundo experimento teve como objetivo demonstrar a eficiência do OS-SLAM em ambientes maiores. Para fazer isso, foi mapeado todo o corredor onde nove laboratórios de pesquisa estão localizados no terceiro andar do prédio. O robô mapeou o corredor principal, o laboratório LaSER, um banheiro e uma pequena parte de um corredor ao lado, como demonstrado na Fig. 5.4. A parte final do corredor lateral é uma área aberta sujeita à forte luz do sol e, portanto, não poderia ser mapeada usando o sensor Kinect.

Para avaliar a abordagem Os-SLAM em uma área maior, a Fig. 5.5 apresenta o mapa final, como resultado desta técnica. O mapa final demonstrou uma estabilidade aceitável com baixas distorções nas paredes do corredor. O vídeo de Nascimento et. al. [25] também apresenta esta experiência na sua segunda parte. O robô começa no ponto de partida (Fig 5.4.) do corredor, entra no laboratório LaSER e o mapeia todo. Em seguida, vai para o corredor lateral, entra no banheiro e termina, depois de mapear o resto da parte do corredor adjacente. Durante esta experiência, uma pessoa é filmada assistindo, enquanto o robô passa pelo corredor. A pessoa é mapeada e, também, removida mesmo ela estando em movimento.

Finalmente, Fig 5.6 e na tabela 5.3., temos os mesmos gráficos e dados já descritos em testes anteriores aplicados neste cenário atual. Note que mesmo num ambiente maior e, con-

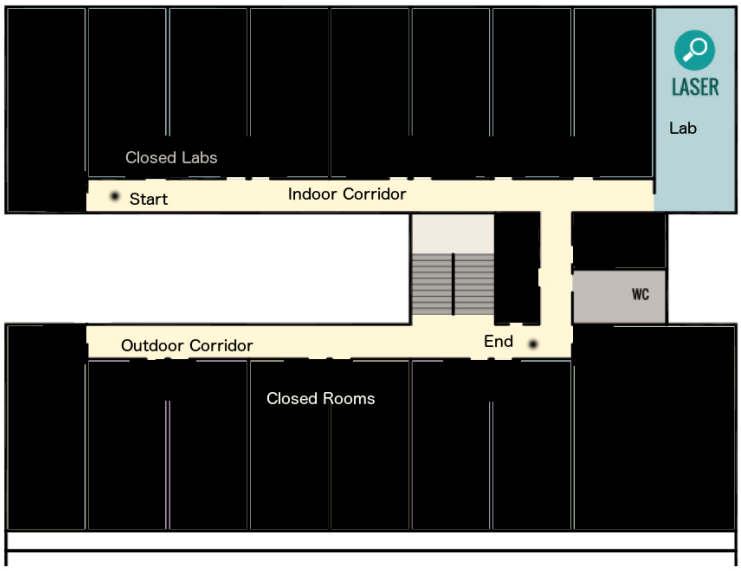

Figura 5.4: Planta do Laborátorio e Corredores Mapeados

Figura 5.5: Mapa criado pelo OS-Map do Corredor e do laboratório

sequentemente, com uma maior quantidade de nós indexados (1,014 nós), não há grande mudança no tempo de processamento de cada fase. Como, por exemplo, no primeiro expe-

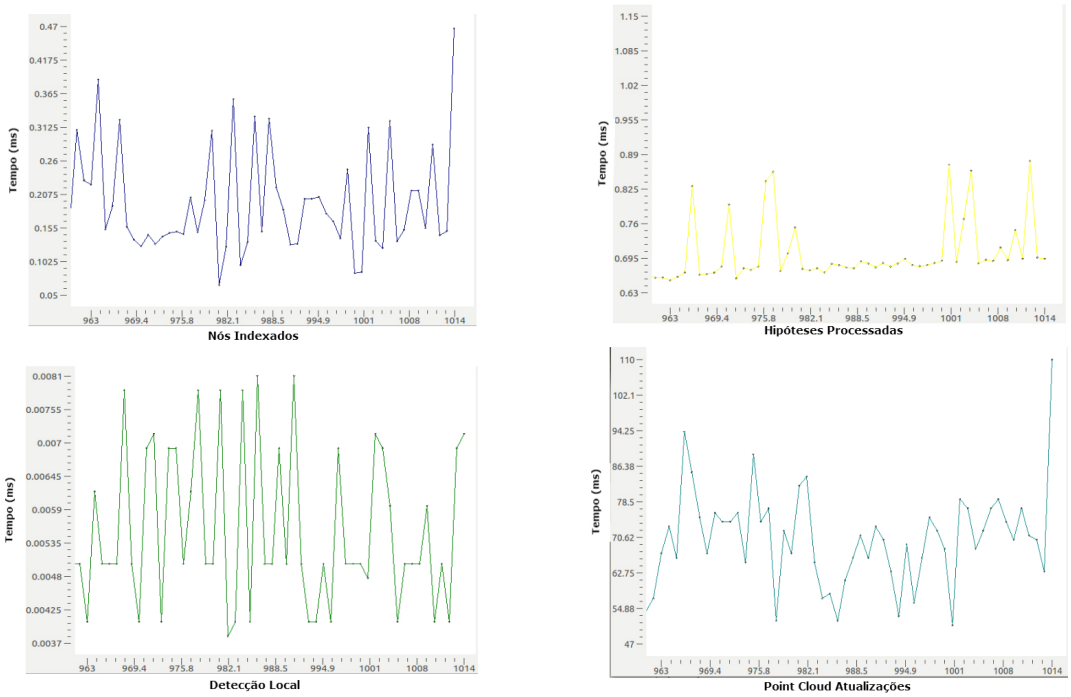

Figura 5.6: Estatistísticas de tempo do OS-SLAM no Experimento do Ambiente Maior

Tabela 5.3: Detalhamento dos Resultados Obtidos no Experimento do Ambiente Grande

Parâmetros	**Detecção Local**	**Nós Indexados**	**Hipóteses Processadas**	**Pc Atualizações**
Nós	1014	1014	1014	1014
Máx (ms)	0.0081	0.47	1.15	110
Min (ms)	0.0037	0.075	0.63	47
Mediana (ms)	0.0059	0.26	0.695	78.5

rimento foi obtido um tempo de 0.22 ms, enquanto que no segundo, um tempo de 0.47 ms para indexação dos nós. Além disso, o Detecção de local (canto inferior esquerdo) teve um aumento menor em tempo que variou de 0.0079ms para 0.0081ms neste cenário maior.

5.1.3 Resutados do Open Dataset (Ground-Truth)

Para uma avaliação quantitativa, aplicada a OS-SLAM, o sistema proposto pela Sturm et. al. [26] que apresentou um algoritmo para validação de sistemas SLAM RGB-D, contém um conjunto de sequências de imagens feitas a partir de um dispositivo RGB-D Microsoft Kinect e unidas por um sistema de captura de movimento. Este conjunto de dados aberto

oferece poses de câmera e verdades terrestres precisas. Os experimentos realizados nesta subseção para testar o desempenho do OS-SLAM usou o Freiburg longo escritório doméstico "$FR3/longo$"e mesa "$fr2/mesa$"conjuntos de dados (ver Fig. 5.7) .

A trajetória e o nível de *groundtruth* foram obtidos por meio de oito câmeras de alta velocidade com uma frequência de 100 Hertz. O trabalho de Sturm et. al. [26] é composto por trinta e nove sequências, cobrindo uma ampla variedade de cenários e movimentos de câmera, que vão desde cenas lentas destinadas para depuração, à cenas longas para avaliação de trajetórias geradas que podem ter a opção de fechamento de loop ou não. Tabelas 5.4 e 5.5 resumem o erro absoluto de Trajetória (ATE) tal como proposto pelo Sturm et al. [26] . ATE computa a diferença absoluta entre o groundtruth e poses estimadas após o alinhamento. Tabela 5.4 apresenta o erro Root Mean Square (RMSE), a mediana e o Max ATE para o desempenho do OS-SLAM em comparação com os resultados do conjunto de dados "$FR3/longo$"usado como groundtruth. Esta comparação entre os valores do OS-SLAM e a trajectória realizada pela nossa abordagem apresentada na Fig.5.7 (parte superior) mostra uma diferença muito pequena entre a abordagem e a *grountruth*, e, portanto, sua precisão.

Tabela 5.4: Comparação Detalhada dos Resultadoss Obtidos do "fr3/long"Dataset

Parâmetros	**OS-SLAM**	**Groundtruth [27]**
ATE RMSE	0.0586 m	0.0329 m
ATE MEDIAN	0.0564 m	0.0297 m
ATE MAX	0.0948 m	0.0698 m

Além disso, a tabela 5.5 apresenta o erro Root Mean Square (RMSE), a mediana e o Max ATE para o desempenho do OS-SLAM em comparação com os resultados do conjunto de dados "$FR2/mesa$". Os valores de verdade de terrestres são comparados com os valores do OS-SLAM e aos valores do SLAM RGB-D [7]. O percurso realizado pela nossa abordagem é também apresentada na Fig.5.7 (parte inferior) e apresenta uma melhoria de 10% na precisão da mediana em comparação com a abordagem SLAM RGB-D.

Tabela 5.5: Comparação Detalhada dos Resultados Obtidos do "$fr2/desk$"Dataset

Parâmetros	**OS-SLAM**	**RGB-D SLAM [7]**	**Groundtruth [28]**
ATE RMSE	0.0557 m	0.057 m	0.0376 m
ATE MEDIAN	0.0493 m	0.053 m	0.0315 m
ATE MAX	0.0961 m	0.099 m	0.0879 m
FPS	30.0 Hz	7.34 Hz	30.0 Hz

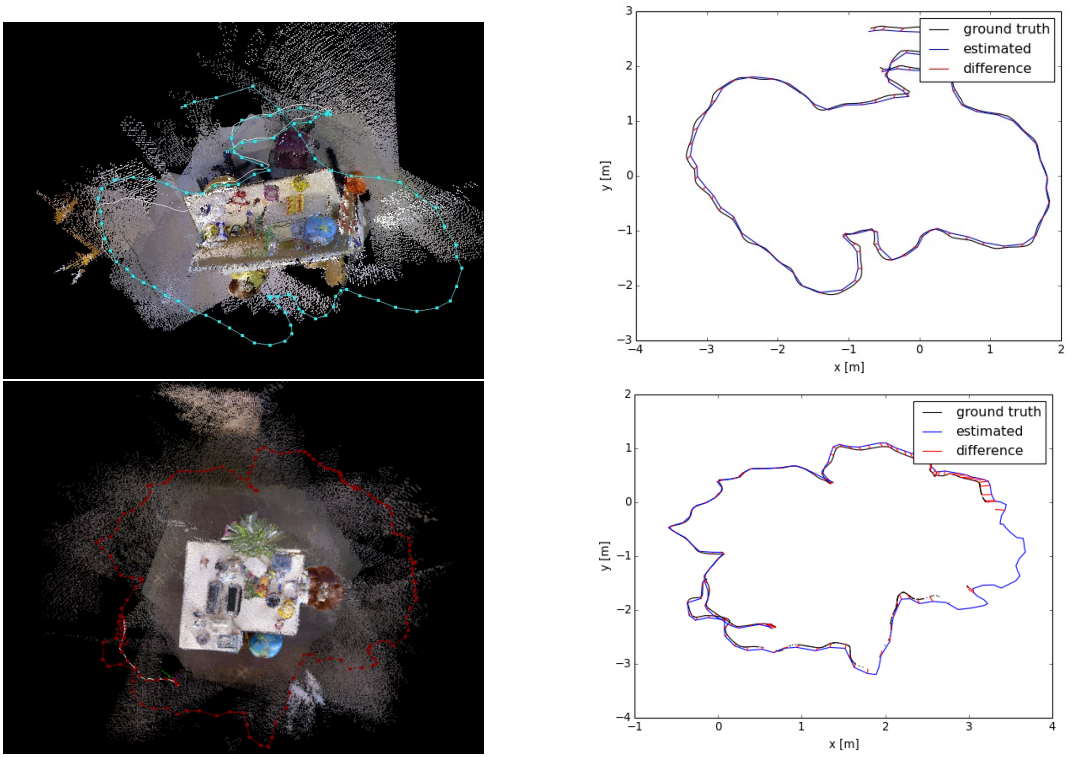

Figura 5.7: Groundtruth long office $fr3/long$ dataset Nuvem de Pontos Final (top left) e Comparação de Trajetória (topo direta). Groundtruth desk $fr2/desk$ dataset Nuvem de Pontos Final (abaixo esquerda) e Comparação de Trajetória (abaixo direta).

5.1.4 Mapeamento 3D e Merging do Ambiente do Quarto

No terceiro experimento, foi utilizado o mesmo hardware mostrado na experiência acima para o mapeamento 3D em cada robô. Em seguida, foi utilizado um servidor para a fusão de mapas. O servidor é um Notebook Alienware rodando ROS e Ubuntu 12:04: Intel (R) Núcleo (TM) i7-3610QM CPU @ 2.30Ghz, 256 GB SSD de 16 GB de HD e RAM. O quarto é 2,00m x 4,50m onde foram adicionados objetos de tamanhos diferentes para demonstrar

a subtração e fusão deles. Dois Turtlebots mapearam a área e geraram seus próprios mapas e trajetórias como também mostrado na Fig.5.8. Após o mapeamento indivídual, as bases foram transferidas para o servidor para o processo de fusão.

Na figura Fig 5.8 é mostrado, abaixo, o quarto usado para executar os experimentos (duas primeiras figuras do topo). O mapa 3D, gerado por cada robô, e o mapa global, após a aplicação do merging, são mostrados nas duas figuras de fundo. No experimento do ambiente, foram utilizados dois robôs móveis para mapeá-lo e alguns objetos móveis para demonstrar a capacidade de identificar e subtrair esses objetos. As áreas em azul e vermelho, respectivamente, fazem referência aos mapas feitos por diferentes robôs. Também pode ser visto o percurso de cada robô e seus mapeamentos individuais. Tal como discutido anteriormente, no mapa global é possível ver uma área que foi revisitada por ambos robôs, a fim de aplicar o processo de fusão. Finalmente, pode-se notar que o mapa não adquiriu distorções pelo processo de fusão.

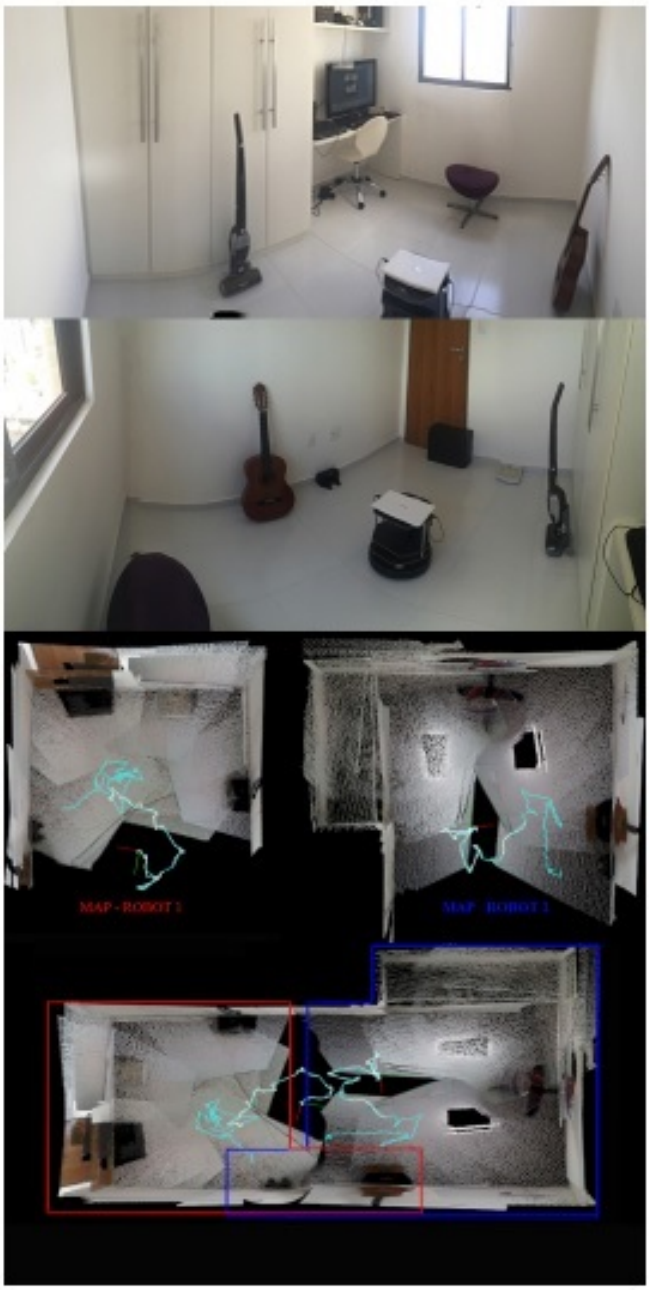

Figura 5.8: Mapeamento do Quarto e Merging

Capítulo 6

Conclusão

Em algoritmos de SLAM 3D, um mapeamento confiável é crucial para um mapa preciso, que por sua vez resulta em um robô bem localizado. A abordagem proposta neste trabalho visa à criação de um novo método de SLAM capaz de aplicar a subtração de objetos e um tratamento de planos, usando conceitos de Planar SLAM. Durante o SLAM realizado em quatro experimentos, a abordagem identificou objetos e os excluiu do mapa final. Os objetos retirados ou foram estáticos (móveis ou não) ou dinâmicos (de pessoas). Os modelos de planos foram extraídos a partir do ponto de turvação original, deixando todos os outros pontos não relacionados com planos, neste caso, os objetos.

A abordagem proposta corrige as distorções no mapa final, como paredes distorcidas (não retilíneas), bem como planos paralelos ou incompletos. Essas distorções são geralmente adicionadas pelos objetos na cena durante a etapa de mapeamento. Nos demais algoritmos encontrados no estado-da-arte, os objetos são considerados no mapa. No algoritmo OS-SLAM, apresentado nesse trabalho, o foco é voltado para os planos de pisos e paredes. Todas as modificações reduziram o custo computacional, demonstrado pelo baixo tempo de processamento durante o SLAM. Finalmente, um mapa preciso e limpo foi adquirido como resultado desta abordagem OS-SLAM.

6.1 Trabalhos Futuros

Em trabalhos futuros pretende-se melhorar a abordagem OS-SLAM através da aplicação de diferentes métodos de identificação de objeto, a fim de aumentar a eficiência do algo-

ritmo. Pois, como apresentado neste trabalho, a abordagem tem a capacidade de identificar e subtrair os objetos do mapa ao decorrer do mapeamento. Em um trabalho futuro anseia-se adicionar, além da identificação, a classificação desses objetos. Podendo, assim, em tempo real, observar que está diante de uma cadeira ou de uma mesa, dentre outros objetos. Alguns dos métodos mais prováveis para este trabalho seria o uso de redes neurais e algoritmos de identificação de marcas naturais.

Outra modificação será realizar uma melhora na etapa de otimização do gráfico. De modo que pretende-se usar algoritmos de otimização heurísticos, que além de classificar e filtrar os planos, como já é feito nesta pesquisa, pretende-se aplicar um descrito capaz de melhorar o desempenho nas demais etapas do algoritmo.

Também visa-se adicionar, na rotina de fusão de mapas já apresentada, características de SLAM distribuído. Possibilitando, assim, a troca de informações em tempo real entre os robôs que estão mapeando como, por exemplo, o compartilhamento de localização atual, dentre outras informações, e também a fusão dos mapas ao decorrer do mapeamento.

Bibliografia

[1] T. P. Nascimento, A. G. S. Conceição, A. P. Moreira, "Multi-robot nonlinear model predictive formation control: Moving target and target absence,"Robotics and Autonomous Systems (Print), v. 61, pp. 1502-1515, 2013.

[2] M. Pinto, A. P. Moreira, A. Matos, H. Sobreira, and F. Santos, "Fast 3D Map Matching Localisation Algorithm,"Journal of Automation and Control Engineering, Vol.1, No.2, pp. 110-114, June 2013.

[3] S. Thrun, W. Burgard and D. Fox, "Probabilistic robotics,"MIT Press, 2005.

[4] R. Ouellette and K. Hirasawa, "A comparison of SLAM implementations for indoor mobile robots,"in Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, pp.1479-1484, 2007.

[5] W. Yong-kun, H. Ju and W. Xing-shun, "A real-time robotic indoor 3D mapping system using duel 2D laser range finders,"in Control Conference (CCC), 2014 33rd Chinese, pp.8542-8546, Nanjing, China, 2014.

[6] A. Ahmad, G. D. Tipaldi, P. Lima and W. Burgard, "Cooperative Robot Localization and Target Tracking based on Least Squares Minimization,"in 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 5696-5701, Karlsruhe, Germany, 2013.

[7] F. Endres, J. Hess, J. Sturm, D. Cremers and W. Burgard, "3D Mapping with an RGB-D Camera,"IEEE TRANSACTIONS ON ROBOTICS, Vol. 30, No. 1, February 2014.

[8] Ros.org Press Release. URL: http://www.ros.org/news/2014/09/sv-ross-team-maxed-

out-wins-first-place-at-the-iros2014-microsoft-kinect-challenge.html. Accessed in February 22nd, 2015.

[9] M. Labbé and F. Michaud, "Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM,"Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pp.2661-2666, 2014.

[10] C. Kerl, J. Sturm and D. Cremers, "Dense Visual SLAM for RGB-D Cameras,"In Proc. of the Int. Conf. on Intelligent Robot Systems (IROS), 2013.

[11] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers and W. Burgard, "An Evaluation of the RGB-D SLAM System,"Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), pp.1691-1696, 2012.

[12] Liang Zhao, Shoudong Huang and Gamini Dissanayake, "Linear SLAM: A Linear Solution to the Feature-based and Pose Graph SLAM based on Submap Joining,"Accepted by IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp.24-30, 2013.

[13] M. Labbé and F. Michaud, "Memory management for real-time appearance-based loop closure detection,"in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1271-1276, 2011.

[14] M. Labbé and F. Michaud, "Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation,"in IEEE Transactions on Robotics, vol. 29, no. 3, pp. 734-745, 2013.

[15] Tae-kyeong Lee, S. Lim, S. Lee, S. An, and Se-young Oh, "Indoor Mapping Using Planes Extracted from Noisy RGB-D Sensors,"in IEEE/RSJ International Conference on Intelligent Robots and Systems October 7-12, 2012. Vilamoura, Algarve, Portugal.

[16] R.F. Salas-Moreno, B. Glocken, P. H. J. Kelly, A. J. Davison. "Dense planar SLAM,"in Proceedings of the 2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR),pp. 157-164, Munich, 2014. DOI: 10.1109/ISMAR.2014.6948422.

[17] Lingni Ma, T. Whelan, E. Bondarev, P. H. N. de With, J. McDonald. "Planar simplification and texturing of dense point cloud maps,"in Proceedings of the 2013 European Conference on Mobile Robots (ECMR), pp. 164-171, Bracelona, 2013. DOI: 10.1109/ECMR.2013.6698837.

[18] Y. Taguchi , Y. Jian , S. Ramalingam , and C. Feng, "Point-Plane SLAM for Hand-Held 3D Sensors, "Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 2013.

[19] M. Kaess, "Simultaneous Localization and Mapping with Infinite Planes,"Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 2015.

[20] R. Cabral and Y. Furukawa, "Piecewise Planar and Compact Floorplan Reconstruction from Images,"in Premier Annual Computer Vision Event (CVPR), 2014.

[21] X. Gao and T. Zhang, "Robust RGB-D simultaneous localization and mapping using planar point features,"in Robotics and Autonomous Systems 72,pp. 1-14, 2015.

[22] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli and M. Vincze, "Point Cloud Library: Three-Dimensional Object Recognition and 6 DoF Pose Estimation,"in IEEE ROBOTICS & AUTOMATION MAGAZINE, no. 9, pp. 80-91, September, 2012.

[23] Hossein Mohimani, Massoud Babaie-Zadeh, Christian Jutten, "Fast Sparse Representation based on Smoothed L0 norm", in proceedings of 7th International Conference on Independent Component Analysis and Signal Separation (ICA2007), LNCS 4666, September 2007, London, pp.389-396.

[24] T. P. Nascimento, "A Novel Approach for Planar RGB-D SLAM Based on Object Subtraction: The Lab Experiment". YouTube, Channel Tiago Nascimento, 2016, URL: https://youtu.be/JmWKhgSiwwI

[25] T. P. Nascimento, "A Novel Approach for Planar RGB-D SLAM Based on Object Subtraction: The Floor Experiment". YouTube, Channel Tiago Nascimento, 2016, URL: https://youtu.be/b-VQzs0KzUs

[26] J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems,"In Proc. of the International Conference on Intelligent Robot Systems (IROS), 2012.

[27] Whelan, T., M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald. “Real-Time Large-Scale Dense RGB-D SLAM with Volumetric Fusion.” The International Journal of Robotics Research 34, no. 4–5 (April 1, 2015): 598–626.

[28] Whelan, T., Johannsson, H., Kaess, M., Leonard, J., and McDonald, J. (2013a). "Robust real-time visual odometry for dense RGB-D mapping."In IEEE Intl. Conf. on Robotics and Automation, ICRA, Karlsruhe, Germany.

[29] C. Hertzberg, R. Wagner, O. Birbach, T. Hammer, and U. Frese. Experiences in Building a Visual Slam System From Open Source Components. In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pages 2644–2651. IEEE, 2011. Cited on page 3.

[30] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard. "Large scale graph-based slam using aerial images as prior information". Autonomous Robots, 30(1):25–39, 2011b. Cited on page 2.

[31] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pages 1–4. IEEE, 2011. Cited on page 3.

[32] Turtlebot Press Release. URL: http://www.turtlebot.com/. Accessed in January 15nd, 2015.

[33] P. Henry, M. Krainin, E. Herbst, X. feng Ren and D. Fox. "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments". The International Journal of Robotics Research, 1–17, 2012.

[34] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. "Speeded-up robust features (SURF)". Comput. Vis. Image Underst., 110:346–359, 2008.

[35] M. Fischler and R. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” Commun. ACM, vol. 24, no. 6, pp. 381–395, 1981.

[36] C. Kerl, J. Sturm, and D. Cremers, “Robust odometry estimation for RGB-D cameras,” in IEEE Intl. Conf. on Robotics and Automation (ICRA), 2013.

[37] N. Fioraio and K. Konolige. “Realtime visual and point cloud slam, ” In Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf. (RSS), 2011.

[38] F. Steinbrucker, J. Sturm, and D. Cremers, “Real-time visual odometry from dense RGB-D images,” in Workshop on Live Dense Reconstruction with Moving Cameras at the Intl. Conf. on Computer Vision (ICCV), 2011.

[39] Leonardo A. Souto, Tiago P. Nascimento,“Distributed Object Subtraction 3D Mapping,” in Robotics Symposium (LARS) and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR), 2015 12th Latin American.